ASTERISK | TRIXBOX | ADHEARSION | VOIP | SELINUX | SIP | NAT

LINUX JOURNAL

Since 1994: The Original Magazine of the Linux Community

An MPD-Based Audio Appliance

Linux and Asterisk Do Telephony

- ASTERISK FOR CONVENIENT PHONE CALLS
- WIRESHARK SNIFFS VOIP PROBLEMS
- VOIP THROUGH NAT
- EMBEDDED ASTERISK
- ADHEARSION WITH ASTERISK
- COMMUNIGATE PRO AND VOIP

MARCH 2007 | ISSUE 155
www.linuxjournal.com
BelltownMedia
USA $5.00
CAN $6.50
0 71486 03102 4 03

PLUS: The D Language

Enterprise and High-Performance Computing Under Your Control

APPRO
Cluster Solutions

Industry Leading 4P x86 Computing

Innovative server technology with outstanding performance and memory scalability

4-Way XtremeWorkstation™

- AMD Opteron™ processors
- Featuring AMD Socket F (1207)
- XGi Volari Z9 graphics chip
- Up to 128GB of DDR2 533/667 memory
- Up to 6.0TB SATA or 2.4TB SAS
- 2 PCIe x16, 1 PCIe x4 and 3 PCI-X
- Redundant fans
- Hot-swappable drives
- Windows® or Linux OS

4-Way 3U XtremeServer™

- AMD Opteron™ processors
- Featuring AMD Socket F (1207)
- Up to 128GB of DDR2 533/667 memory
- Up to 4.5TB SATA or 1.8TB SAS
- 3 PCI-X and 2 PCI Express x16
- Redundant power supplies and fans
- Hot-swappable drives
- ServerDome Management – IPMI 2.0
- Windows® or Linux OS

Leveraging Xen Virtualization with the Appro XtremeServer

Go to http://www.appro.com/whitepaper/White Papers.asp

AMD Opteron™ Processors:
- Quad-Core Ready - increase capacity without altering datacenter infrastructure
- Best performance per-watt with energy-efficient DDR2
- Optimized system performance with Direct Connect Architecture

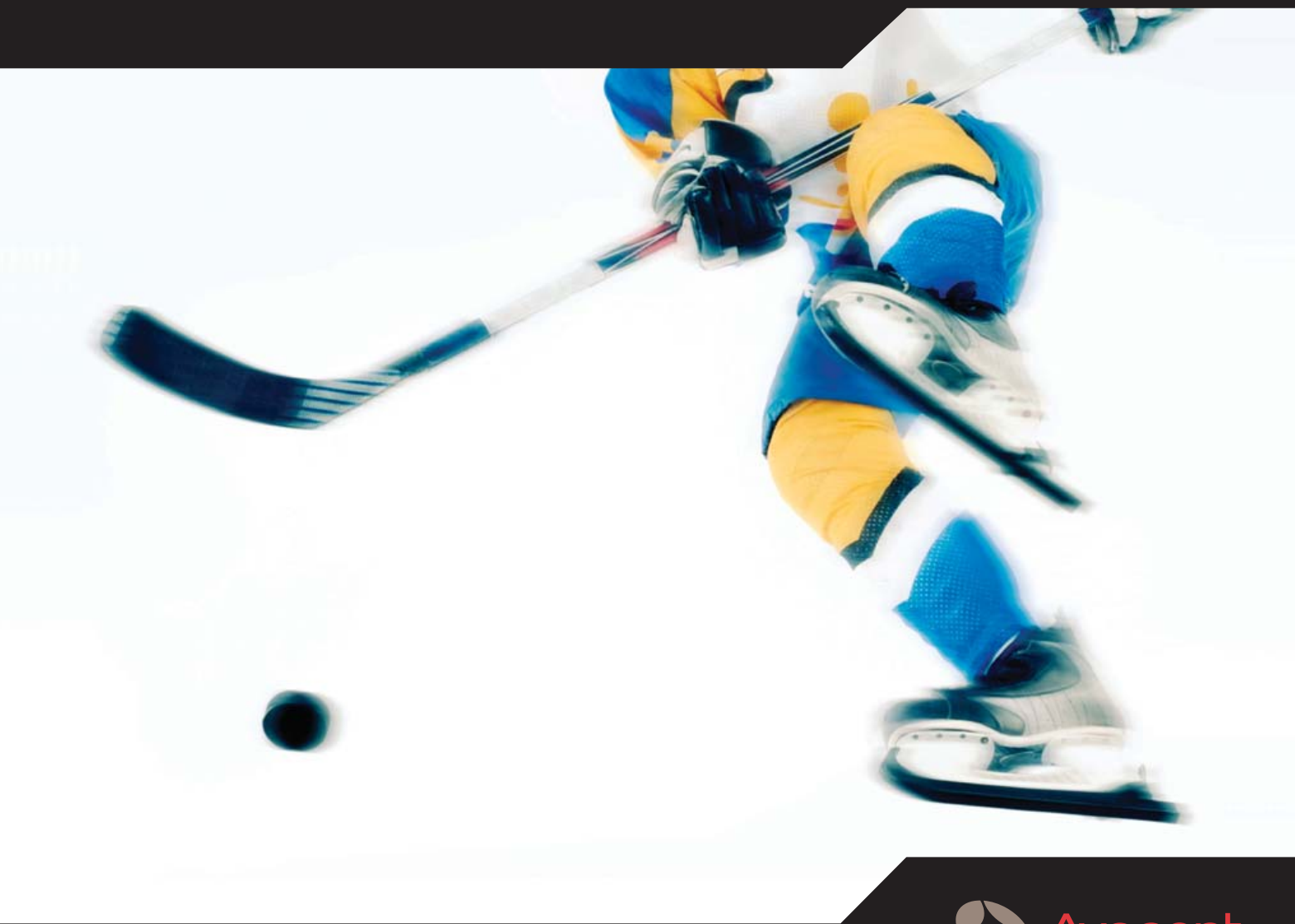
For more information, please visit www.appro.com or call Appro Sales at 800-927-5464 or 408-941-8100.

AMD 64 Opteron™

Today, Dan configured a switch in London, rebooted servers in Sydney, and watched his team score the winning goal in St. Louis.

With Avocent data center solutions, the world can finally revolve around you. Avocent puts secure access and control right at your finger tips – from multi-platform servers to network routers, your local data center to branch offices, across the hall or around the globe. Let others roll crash carts to troubleshoot – with Avocent, trouble is on ice.

To learn more, visit us at www.avocent.com/ice to download Data Center Control: Guidelines to Achieve Centralized Management whitepaper or call 866.277.1924 for a demo today.



Avocent®
The Power of Being There®

CONTENTS

MARCH 2007
Issue 155

ILLUSTRATION ©ISTOCKPHOTO.COM/STEFAN WEHRMANN

FEATURES

50 Time-Zone Processing with Asterisk, Part I
Hello, this is your unwanted wake-up call.
Matthew Gast

56 Home Box to Trixbox
Add a digital receptionist to your home.
Michael George

66 How to Configure SIP and NAT
Swatting NAT for VoIP.
Sean Walberg

70 Expose VoIP Problems with Wireshark
Shark in the network.
Sean Walberg

ON THE COVER

- **An MPD-Based Audio Appliance, *p. 86***
- **Asterisk for Convenient Phone Calls, *p. 50***
- **Wireshark Sniffs VoIP Problems, *p. 70***
- **VoIP through NAT, *p. 66***
- **Embedded Asterisk, *p. 78***
- **Adhearsion with Asterisk, *p. 74***
- **CommuniGate Pro and VoIP, *p. 82***
- **The D Language, *p. 90***

The competition doesn't stand a chance.

If you base deployment decisions on performance and price, Coyote Point's for you. We've cornered that market.

To prove it we asked The Tolly Group to evaluate our E350si application traffic manager against the competition. The results speak for themselves.

Throughput? Almost 40% more than others in our space. Cost of transactions per second? Up to four times less. Connection rate? In some cases, one-sixth the cost. One-sixth! And we're told Coyote Point is the #1 choice for today's open source networks.

But don't just take our word for it. Get the facts. Call 1.877.367.2696 or write info@coyotepoint.com for your free copy of the full Tolly Report.

CoyotePoint
Systems Inc

CONTENTS MARCH 2007 Issue 155

COLUMNS

18 REUVEN M. LERNER'S AT THE FORGE
Dojo

22 MARCEL GAGNÉ'S COOKING WITH LINUX
Free Long Distance—Really!

28 DAVE TAYLOR'S WORK THE SHELL
Compact Code and Cron Contraptions

30 MICK BAUER'S PARANOID PENGUIN
Introduction to SELinux, Part II

34 JON "MADDOG" HALL'S BEACHHEAD
Waysmall

36 DOC SEARLS' LINUX FOR SUITS
DIY Internet Infrastructure

96 NICHOLAS PETRELEY'S /VAR/OPINION
Dealing with the Devil

QUICK TAKES

48 COYOTE POINT EQUALIZER E550SI LOAD BALANCER
Logan G. Harbaugh

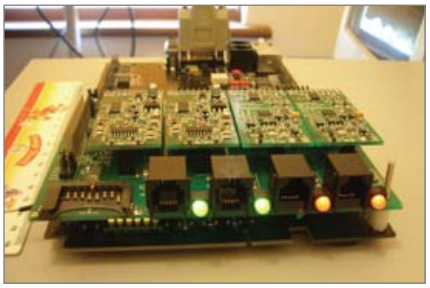
IN EVERY ISSUE

8 LETTERS
12 UPFRONT
42 TECH TIPS
46 NEW PRODUCTS
81 ADVERTISERS INDEX

INDEPTH

74 APPLYING ADHEARSION TO ASTERISK
A gem of an Asterisk tool.
Jay Phillips

78 COMBINE UCLINUX AND ASTERISK ON A CARD
uClinux is the Digi-Key to embedding Asterisk.
David Rowe



82 VOIP WITH COMMUNIGATE PRO
How to do VoIP with CommuniGate Pro.
Daniel Sadowski and Stephen Pratt

86 BUILDING A MULTI-ROOM DIGITAL MUSIC SYSTEM
MPD on simple hardware goes a long way.
Chad Files

90 THE D PROGRAMMING LANGUAGE
What comes after C++ and C#?
Ameer Armaly

48 COYOTE POINT EQUALIZER E550SI

Next Month

SECURITY

You're running Linux but does that mean your systems are as secure as they need to be? Next month, we'll help you find out and give you what you need to lock down and manage your Linux environments. We'll tell you why Single Packet Authorization (SPA) is a step up from port knocking, and then we'll follow up in the next issue with instructions on how to implement SPA. How does your filesystem security stack up? It stacks nicely with eCryptfs, as you'll see. We'll take you under the hood of Multi-Category SELinux and explain the details behind OpenSSH.

As always, there's much more. We'll show you how to create dynamic forms and reports with Inkscape and XSLT, and follow up with more information about time-zone processing with Asterisk. We'll tell you why MySQL deserves a double-take, and Marcel Gagné will explore Mondo Rescue disaster recovery.

USPS *LINUX JOURNAL* (ISSN 1075-3583) is published monthly by Belltown Media, Inc., 2211 Norfolk, Ste 514, Houston, TX 77098 USA. Periodicals postage paid at Houston, Texas and at additional mailing offices. Cover price is $5 US. Subscription rate is $25/year in the United States, $32 in Canada and Mexico, $62 elsewhere. POSTMASTER: Please send address changes to *Linux Journal*, PO Box 980985, Houston, TX 77098. Subscriptions start with the next issue.

Gemini 2U

– Ultra Dense Series

- Two fully independent systems in a 2U
- Ability to run two discrete operating systems in one box
- Up to 16 CPU cores
- Up to 12 hot-swappable SATA, SCSI, or SAS hard drives
- RAID 0, 1, 5, 6, 10, 50 available on both systems
- Opteron™ or Xeon™ multi-core processors
- Up to 64GB memory per motherboard
- One available PCI-X or PCI-E slot per motherboard
- High efficiency AC and DC power options

2 Systems in One

Built on open standards, the Gemini 2U elegantly accommodates two discrete motherboards in a 25" chassis uniquely designed for easy access from the rear.

Gemini 2U represents the realization of intoxicating power and superior environmental specifications, with considerably less power consumption, less heat and less noise. Remarkably, it all fits nicely into any standard rack. Front to back, Gemini 2U is both powerful and efficient.

At Open Source Systems we understand you need practical, customizable, and affordable solutions that are easy to manage and maintain.

For more information and to request your evaluation unit today, visit us at **www.OpenSourceSystems.com**, or call direct at 866.664.7867.

Patent Pending

AMD Solution Provider PLATINUM

AMD 64 Opteron

Open Source Systems™

LINUX JOURNAL

Linux Journal **is published by, and is a registered trade name of, Belltown Media, Inc.**
PO Box 980985, Houston, TX 77098 USA

TYANPSC™

Hands On Supercomputing

Typhoon™ 600 Series Personal Supercomputer

TyanPSC's Typhoon™, the next generation turnkey Personal Supercomputer has the power to blow away all your computational needs! Purpose-built for office and laboratory environments, easy to deploy and use, the Typhoon™ provides intense computational power in remote or constrained places, works like a PC and is whisper quiet.

High Performance Computing Just Got Cooler

Clusters of Typhoons / Low Power
Small size Form Factor / Under Mobility

Typhoon

T-630 DX / T-650 QX Series

- Up to 186 / 256 GFlops at your desk!
- Turnkey, Easy-to-Deploy, and Easy-to-Use
- Integrated 5 node cluster - up to 20 / 40 processor cores in a box!
- Plugs into standard wall outlet - only uses 15 Amps
- Microsoft® Windows® Compute Cluster Server 2003 pre-installed
- High Performance in Constrained spaces: Office, Remote, Plane, Boat, etc
- RAID capable

intel Xeon inside™

Microsoft® Windows®

Windows Compute Cluster Server 2003

TYANPSC™ Personal Supercomputer

Tyan Computer USA
3288 Laurelview Court
Fremont, CA 94538 USA
Tel: +1-510-651-8868 Fax: +1-510-651-7688
Pre-Sales Tel: +1-510-651-8868 x5120
Email: marketing@tyan.com

For More Information, please visit
www.tyanpsc.com

letters

Gnull and Voyd

Just when I thought you could not possibly come up with a column more irritating than Marcel Gagné's Cooking with Linux (full of useful information embedded in silliness), you bring us Gnull and Voyd. Give us the tech tips; please lose the trailer trash schtick.

--

Patrick Wiseman

As you (and others) wish. Tech tips stay; Gnull and Voyd go.—Ed.

!Gnull and Voyd

Keep up the good work. I enjoy your /var/opinion column, and I continue to look for Reuven Lerner's fine columns. You folks have helped me become a better system administrator over the years, and I thank you for that.

--

David MacDougall

Forgiveness

A few months ago, I wrote to you saying we had to break up—you had become too chatty and opinion-filled for my taste. I have to admit that it turns out I was bluffing. I love you too much, and today, I renewed my subscription.

You did take a significant dip in quality, but things have gotten better the last couple of months. More important, though, is the fact that you started from such a high-quality base that even when it dipped, you were still the only Linux publication for me.

--

Andy Balaam

Organized Repression

Your /var/opinion [January 2007] regarding the trade-offs between GPLs versions 2 and 3, reminded me of a wry remark that's worth repeating: "There's no such thing as freedom—it's just a question of how the repression is organised!"

--

Struan Bartlett

More Than Just Developers

You claim that "the only people who are truly harmed by having the software on ROM are the tiny minority of hackers who want to run a modified version of the software on the gadget" [January 2007 /var/opinion]. This statement is false. Hackers may be the only people whose goals are *directly* impeded by immutable embedded software. But where does the software created by hackers eventually trickle down to? The user, who would know no more about the software than how to load it into the device and make use of whatever enhanced features it provides. The "vast majority" of users are harmed by the chilling effect on a "tiny minority" of capable developers because they do not benefit from the software that otherwise would have been developed.

--

Ryan Underwood

Point taken. But if the source code is available, as it must be under GPLv2, then developers can learn from it and use it, just not on a particular device.—Ed.

DocBook XML and CSS

David Lynch's November 2006 article on using DocBook XML to build simple Web sites was timely for me. For many years, I'd been writing the documentation for my open-source projects in raw HTML, but I've recently "seen the light" and now use a combination of DocBook XML and CSS. However, my deployment approach is quite different from David's. Instead of relying on frames and the browser's ability to convert XML to HTML—and suffer from the complications this brings—I simply convert the DocBook XML to HTML off-line, then upload the output to my Web site. This is a much simpler approach that still preserves the key advantages of DocBook. I recommend it to anyone writing software manuals or simple Web sites, looking for a clean split between content and presentation.

For an example of how it's done, download the HR-XSL Project (**hr-xsl.sourceforge.net**), and look in the doc directory.

--

Trevor Harmon

Ode to Joy

Jon Hall is making some extremely weak arguments against patents [Beachhead, January 2007]. First and foremost, the argument should not be *if* we should have software patents. The argument should be on how software patents and patents in general are implemented and maintained.

Although it may take me several years to come up with a completely new operating system, it may take someone else only a few weeks or months. This does not mean that this new, novel operating system should not be patented and protected so that big companies cannot steal it.

You see, to invent something, the inventor is usually intimately involved in that field or research. Sometimes the best ideas just appear out of nowhere. The idea itself may be easy or hard to implement, it may require more or less time, but what matters in the end is the ingenuity and usefulness.

This is one thing everyone who is complaining about patents is missing. *Patents are there to protect the small guy.* It is not the other way around. It may look like that today, as the implementation and enforcement of the patent laws may be unfortunate, but ultimately, the idea behind a patent is to *protect* your invention.

Imagine a world with no copy protection, trademarks, patents or other laws protecting ingenuity and uniqueness. In a short period of time, there would be no invention, no new music or works of art. We would see only repeats and same stuff over and over again. There would be no incentive to innovate. It would simply not be a smart investment to invest in invention. That kind of world would be just terrible.

To some extent, this is already happening with software development. Small shareware developers that used to drive invention and put pressure on big companies are now having very little reason to invent. It is hard to protect the invention, and if they don't

Breaking Numbers Down

I read with interest "Breaking Numbers Down" [Dave Taylor's Work the Shell, *LJ*, December 2006], having recently dealt with this problem myself. I believe bc is a generally overlooked, but very powerful, UNIX utility. Taylor's script nicely illustrates the use of bc, and it is very useful for most purposes, but unfortunately, it doesn't scale well when dealing with the entire range of binary prefixes (**en.wikipedia.org/wiki/Binary_prefixes**).

First, the numeric test -lt used to find the first non-zero {kilo|mega|giga}int fails with numbers larger than 2^{73}-1 (at least on bash-3.0-8.2 running under kernel 2.6.8-24.25-smp).

Second, to extend this to deal with petabytes, exabytes, zettabytes and yottabytes, a total of 16 calls to bc must be employed, with the attendant overhead of shelling out to the system for each.

An alternative, which uses bc more efficiently and avoids testing a number too large for shell, follows:

```
# total letters
nc=`echo -n $1 | wc -c`

# numeric letters
nn=`echo -n $1 | tr -cd '[0-9]' | wc -c`

if [ -z "$1" -o $nc -ne $nn ] ; then
   echo "Usage:  kmgp <integer>"
   echo "         where  0 <= integer <= (2^100)-1 "
   exit 1
fi

SIprefix=" KMGTPEZY" # kilo, mega, giga, tera, peta,
➥exa, zetta, yotta

# what is the closest power of 1024?
# ( ln(1024)=6.93147180559945309417)
order=`echo "scale=0 ; 1 +
➥l($1)/6.93147180559945309417" | bc -l`

# find the letter associated with this power of 1024
letter=`echo "$SIprefix" | cut -c $order`


if [ $nn -gt 3 ]
then scale=3
else scale=0
fi

value=`echo "scale=$scale ; $1/(1024 ^($order-1))"
➥| bc -l`

echo "$value$letter"
```

This version contains two calls to bc and one to cut. The calls to bc merit some discussion: The first:

```
# what is the closest power of 1024?
# ( ln(1024)=6.93147180559945309417)
order=`echo "scale=0 ; 1 +
➥l($1)/6.93147180559945309417" | bc -l`
```

determines the closest power of 2^{10} by using the fact that dividing a logarithm by the logarithm of N is the same as taking its Nth root. The offset by one compensates for the fact that cut is one-based, not zero-based. Note that we are loading bc's math libraries by using bc -l.

The second:

```
value=`echo "scale=$scale ;
➥$1/(1024 ^($order-1))" | bc -l`
```

divides by 1024 raised to the correct order and scales to an appropriate number of decimal places.

Averaging the time taken for both scripts on 400 arbitrary numbers, I find that the logarithm-based script is a little more than three times faster. On the other hand, if numbers larger than several hundred gigabytes are of no interest, the original script is faster.

--

John

protect it, someone bigger will come along and take their market, or if that doesn't happen, a less usable, but free version will be published. Why invent? It's better to steal someone's idea, hire some cheap labor and just put the money into marketing rather than R&D.

On a side note regarding the music and visual art comments Jon made: imagine if I could copy *Ode to Joy*, then add two notes at the end of the piece and claim it as my own. If I could market it better and more strongly than the original composer (Beethoven), who would be able to say who actually wrote that piece of music in the first place (assuming I was living in the same time period)?

More important, if that were to happen to Beethoven, would he be able to write a single piece of music again without being afraid someone will steal it? Would he write music at all?

--

Nebojsa Djogo

Jon "maddog" Hall replies: *I agree entirely that "big companies" (or anyone else) should not be able to steal your work, but I disagree that software patents are the way to make sure that doesn't happen.*

Copyright, contract and licensing law were applied to software a long time before software patents generally became coded into law in 1995. People were protecting their software way before software patents were generally available.

Regarding the big and small point—the small software creator does not have the money to fight a software patent battle in court. Witness the contest between RIM and NTP recently, where three of NTP's four claimed patents were

overturned at a cost of millions of dollars in legal fees. The fourth one might have been overturned, but RIM and NTP decided to "settle". The only people who won from this debacle were the lawyers.

I did not advocate a world without "copy protection", only software patents. I (and most of the Free Software community) appreciate copyrights, trademark and trade secret laws for the protection of people's ingenuity. Free Software relies on copyrights for its licensing.

Regarding the Beethoven scenario—Beethoven would have sued you for copyright infringement and probably would have won in court. But, he would not have been able to block you from using a triplet, or some other "process" of writing music.

*Unfortunately, patents are not foolproof in protecting an invention. Witness the issues around Alexander Graham Bell and Antonio Meucci (***www.italianhistorical.org/MeucciStory.htm***).*

All Beethoven would have had to do was publish his symphony in any public, dated document (a much simpler and less costly procedure than a patent application), and it would have been protected by copyright law.

Thank you for writing your letter, but I stand my ground against software patents.

At the Forge

Reuven's column in *Linux Journal* is one of my favorites, and I read and read it again, but the one in the January 2007 issue is one of the best articles I have ever read in *Linux Journal*. Please offer my thanks to Reuven for his job.

--

Stefano Canepa

Myths?

I appreciated Paul McKenney's article explaining recent advancements in real-time Linux ["SMP and Embedded Real Time", January 2007], and I especially enjoyed the "priority boost" comic strip in Figure 13. However, I was a bit disappointed at his attempts to dispel certain "myths" about parallel and real-time programming.

In Myth #2, for instance, I was hoping for some insight as to why parallel programming is not "mind crushingly difficult". Unfortunately, Dr McKenney's explanation was nothing more than a declaration that "it is really not all that hard". Until I see a more substantial argument to dispel this so-called myth, I'm inclined to believe that parallel programming is in fact quite difficult. To paraphrase Edward A. Lee: insanity is to do the same thing over and over again and expect the results to be different; therefore, programmers of parallel systems must be insane.

Also, in Myth #5, Dr McKenney is propagating a myth instead of dispelling one. He notes that as Web sites become spread across multiple servers—load balancers, firewalls, database servers and so on—the response time of each server must "fall firmly into real-time territory". Here he equates "real time" with "real fast", which is such a common myth, it probably should be at position #1 in his list. In fact, real-time systems are all about predictability, not speed, and their design often sacrifices performance for predictability. And yet, Dr McKenney implies that moving Web servers to a real-time environment will magically reduce their response times. This is simply not true. The response time of a Web server goes up only in the presence of overload—too many people hitting it at once—and when this happens, a real-time Web server will fail just as easily as a non-real-time server would.

I hope that any Web admins out there who have been inspired by Dr McKenney's article will put down their copy of Ingo Molnar's real-time preemption patch and forget about real-time Linux. Simply adding another server behind their load balancer will have a much greater impact in improving overall response time—and require far less effort!

--

Trevor Harmon

Typo

There is a mistake in David Lynch's January 2007 article "How to Port Linux when the Hardware Turns Soft". He says that BSP stands for Broad Support Package. This is incorrect. The correct expansion is Board Support Package.

--

Trevor

Play with fun toys.

Garmin International, the world leader in GPS technology located in Olathe, Kansas, is growing at a rapid rate. We are on the

hunt to find the best and brightest individuals to help us continue creating cutting edge products.

POSITIONS CURRENTLY AVAILABLE INCLUDE:

LINUX SOFTWARE ENGINEER IRC265 This person will be responsible for developing software for Garmin's communication and navigation products. This requires the candidate to be experienced in C, C++ or other selected languages for either embedded or application development for desktop Linux in accordance with Garmin software development methodology. Responsibilities will include testing software using debuggers, emulators, simulators and logic analyzers, performsoftware releases and software quality assurance activities and maintenance on products already in production and new product software design.

The ideal candidate will have a Bachelor of Science degree in Computer Science, Electrical Engineering or Computer Engineering from a 4 year college or university or the above/equivalent education or experience. Excellent academic record of composite GPA of 3.0 or better and experience and/or training in high level languages such as C, C++ for either embedded or desktop Linux. Working knowledge of the GNU tool chain for software development and experience with application frameworks such as Gtk+, Gtkmm, Qt, and Qtopia.

SYSTEMS ADMINISTRATOR - INTERNET IRC385 This person will be primarily responsible for maintaining the infrastructure that powers the Garmin web site. This requires the candidate be experienced supporting Open Source web systems including Linux, Apache, JBoss and MySQL, load balancing devices, security tools, and assisting developers. The candidate must show proven capability to support a highly available web site as well as the underlying infrastructure required to provide stability, security, and scalability on the Internet. This includes intimate knowledge of Bind, LDAP, J2EE, email, ethernet and wide area networking. Security skills and knowledge, and an ability to clearly communicate these concepts and requirements, is essential.

The ideal candidate will have worked 2+ years as a Network or Web engineer supporting a web site that takes millions of hits per day, and includes brochure as well as ecommerce and advanced data delivery functionality. The ability to work with application and site developers to deploy scalable, secure and stable web applications is required. Support for IIS, MS SQL, Windows, and Oracle iAS is needed. Experience operating in a co-located or remote-hosted and global load balanced environment is a plus. Knowledge of emerging Internet technologies and standards is expected. On-call support is required.

For more information or to apply on-line, visit our website at **www.garmin.com/careers.**

GARMIN™

diff -u

WHAT'S NEW IN KERNEL DEVELOPMENT

Promise has released hardware specifications for the full line of chipsets supported by the **sata_promise.c driver**. This is excellent news, especially considering that Promise has been reluctant to release any specs until now. **Jeff Garzik** has been negotiating with them on this issue for a while now, and as this news shows, he has made significant headway. It's great to see Promise begin to turn around on this issue.

Karel Zak is working toward forking the **util-linux project** or taking it over from **Adrian Bunk**. Karel maintains the Red Hat package, and Adrian has not been as active as Karel would like on the project lately. Karel posted to the linux-kernel mailing list recently, explaining his plans to create a git repository and home page for the project and to start merging known bug fixes into the code base. His intention is to make the transition as peaceful as possible and have a good handoff from Adrian. As **H. Peter Anvin** points out, an outright forking of the code could be the best way toward that peaceful transition. As he says, once Adrian sees that Karel is able to do a good job, he might feel less reluctant to let go of the project.

Mikulas Patocka has released an initial version of **SpadFS**, a new filesystem he created as part of his PhD thesis. It attempts to solve the problem of sudden reboots in a simpler way than journalled filesystems. Mikulas finds journalling too complex and bug-prone, preferring a method that he calls crash counting. In this technique, the filesystem keeps track of whether it has been mounted or unmounted; it also tags fresh data with this information, until the data can be written properly into a consistent state. If the computer crashes and comes back up, the filesystem will notice that its saved mount state doesn't match its current mount state, and it can then revert to the most recent consistent state of its data. SpadFS seems to be on the fast track to kernel inclusion, with support from **Linus Torvalds**, who has gone so far as to say that it "doesn't look horrible" to him.

The brand-new **ext4 filesystem** has been accepted into the official kernel. Actually, at its current state of development, it's more just ext3 with some additional patches, making acceptance a much less difficult prospect than other filesystems, notably ReiserFS. But unlike ext3, the ext4 code will continue to accrue new features and undergo other large changes. These large changes were one of the primary reasons why Linus Torvalds insisted that the developers give the project a new name. The filesystem, he said, should be completely trustworthy. Once stabilization occurs in a filesystem, he believes that should be the end of it. Little enhancements and bug fixes might still be okay, but for larger changes involving greater risk of data corruption and other problems, a stable filesystem just should not have to go through that. This was the motivation behind forking ext2 into ext3, and probably will be the motivation behind forking ext4 to ext5 in a few years.

Presenting a system's power source to the user via a consistent interface has not been a high priority in kernel development until recently. Each different type of battery has had its own interface into the kernel, creating a generally hard-to-use mess. But **David Woodhouse** recently announced a **generic battery class driver**, regularizing the entire interface. He is even considering adding an AC power interface to this driver, though there have been some voices of dissent. **Richard Hughes**, for instance, feels that batteries and AC power are sufficiently different to warrant a separate driver for each. But as David currently sees it, the two interfaces will be so similar that there's no point in duplicating a driver for each. So far, the debate has not been decided. Having at least one generic power driver, however, does seem to have a lot of support among the kernel hackers.

—ZACK BROWN

LJ Index, March 2007

1. **Number of journalists in prison, worldwide, on December 7, 2006: 134**
2. **Increase in jailed journalists over one year earlier: 9**
3. **Number of nations with jailed journalists: 24**
4. **Number of jailed journalists that are Internet-based: 67**
5. **Position of China among the world's leader in jailed journalists: 1**
6. **Number of jailed journalists in China: 31**
7. **Firefox's market share percentage in Slovenia: 39**
8. **Firefox's market share percentage in Finland: 35.4**
9. **Firefox's market share percentage in Slovakia: 34.3**
10. **Firefox's market share percentage in Poland: 32.3**
11. **Firefox's market share percentage in the Czech Republic: 31.3**
12. **Firefox's market share growth rate percentage in France: 19.5**
13. **Firefox's market share percentage in North America: 13.5**
14. **Firefox's market share percentage in Oceania: 21.4**
15. **Average minutes and seconds spent on a Web site over mobile phone: 2:53**
16. **Average minutes and seconds spent on a Web site over other connections, including PCs: 5:03**
17. **Linux server revenue in billions of dollars for the last measured quarter: 1.5**
18. **Linux year-over-year revenue growth percentage: 5.4**
19. **Linux share percentage of all server revenue: 11.8**
20. **Reliability rank of Linux-based Tiscali: 1**

Sources: 1–6: Committee to Protect Journalists | 7–16, XiTi Monitor | 17–19: IDC | 20: Netcraft.com

—Doc Searls

Cast Freely

Campware has released Campcaster 1.1, an open-source radio broadcasting system that runs on Linux (Debian and Ubuntu, so far) and is made to scale from individuals to staffed stations to multiple stations in a network. Features include:

- Live, in-studio playout
- Web-based remote station management
- Automation
- Playlists
- Centralized program material archives
- Fast playback (using GStreamer)
- Program sharing
- Search-based backup
- Localization
- Open, extensible architecture (including extensive use of XML-RPC APIs)

Campcaster is the latest open-source product from Campware, an initiative that supports independent news and media organizations in emerging democracies. Other products are Campsite (multilingual news publishing), Cream (customer relationship management or CRM) and Dream (newspaper distribution management). Find links to all and more at **campware.org**.

If you're interested in Campcaster, you also may want to look at Rivendell, a heavy-duty radio broadcast automation system developed by Salem Radio Labs, a division of Salem Communications, which owns one of the largest chains of radio stations in the US. In addition to Rivendell, Salem Radio Labs has a pile of Linux-based open-source products. Many are in use outside the US as well. Find them at **salemradiolabs.com**.

Although Campware is focused on emerging democracies and Salem Radio Labs is focused on the Christian broadcasting community, goods produced by both are wide open for anybody to use.

—DOC SEARLS

A New Province of Open Source

Manitoba is going open source. And vice versa.

The Manitoba Media Centre is a new "Open Source Entertainment Engineering, Innovation, and Production Research Facility" in Winnipeg, capitalized by a $20 million investment from the Provincial Government of Manitoba and Linux Media Arts—a Los Angeles-based media production company. The Centre grows out of a trade mission effort between Manitoba and California.

The Manitoba Media Centre will work on development of multimedia applications for film, television and the Internet. It also will concentrate on the needs of educational institutions and developing economies.

Michael Collins, CEO of Linux Media Arts, says, "Our goal is to leverage the $20 million into at least a $100 million endowment within 18 months to two years through consulting, product development contracts and sponsorships." For Linux specifically, he intends to approach development "from the perspective of what is important to multimedia users. In other words, tool and applications and kernel changes that will improve the media experience. The various distros and the companies who support them do not have this market specifically in mind. It's mainly a support issue. How best can we support the multitudes of users worldwide building systems and products for this $1 trillion market? We think there is room for more."

Find out more at **manitobamediacentre.org**.

—DOC SEARLS

Top Ten Reasons
We're Changing *LJ* Back to Its Original, Smaller Size

10. **The old /var/opinion used to fit perfectly in my birdcage.**
9. **The edges stick out when I'm pretending to read *Playboy*.**
8. **My vintage Logitech hand scanner needs three passes per page.**
7. **It doesn't fit in my hand.**
6. **It slides off the toilet lid.**
5. **Saves trees.**
4. **It doesn't fit in my magazine holder.**
3. ***Wired* did it.**
2. **The extra white space caused too much glare.**
1. **Number one is irrelevant—you don't need *LJ* when doing number one.**

They Said It

Just because it's hard doesn't make it worth doing.
—Britt Blaser

It's often assumed that putting secret codes in music files to protect them from being copied is a way to prevent copyright infringement. But it's not really about that at all. At heart, digital rights management (DRM) is a business strategy, not a police action. And that strategy may be reaching the end of its natural life.
—Nick Carr, www.roughtype.com/archives/2006/12/curtains_for_mu.php

We are moving to a world in the 21st century in which the most important activities that produce occur not in factories and not by individual initiative but in communities held together by software. It is the infrastructural importance of software which is of primary importance in the move to the post industrial economy...Software provides alternative modes of infrastructure and transportation that is crucial in economic terms because the driving force in economic development is always improvement in transportation....Software is creating roadways that bring people who have been far from the center of human social life to the center of that social life....Software can be used to keep software from being owned.

Now we live in a different world for the first time, all the physics, all the mathematics, everything of beauty in music and the visual arts, all of literature can be given to everybody everywhere at essentially zero marginal cost beyond the cost of making the first copy.
—Eben Moglen, keynote at Plone, www.youtube.com/watch?v=NorfgQlEJv8

Tales of an Asterisk Addict

I've been building my Asterisk phone system at home for about a year and a half now. I hope that by reading this article, you get an idea of some of the configurations that can be built with Asterisk and how each configuration functions, as well as each configuration's drawbacks. Asterisk has given me the flexibility to handle incoming calls in ways that you'd never think possible and to make outgoing calls more convenient. For those of you who don't know what Asterisk is, Asterisk is a software program that can interface with the Public Switched Telephone Network, PSTN, and provides voice mail, conferencing and other sophisticated call-handling features, all under your control.

I first got started with Asterisk when I was searching the Web trying to learn what I could about VoIP. When I happened upon the Asterisk program, I couldn't believe it could be everything it was hyped up to be. Once I got it installed and configured, I was able to download software that I could use like a regular telephone, but I could call only other users on my server. This was a fun toy, and my young son got a kick out of hearing his dad's voice on the computer.

As fun as this standalone system was to play with, it wasn't very useful, so I decided to add an inexpensive interface card to the system that would allow it to make and receive phone calls over our regular PSTN phone line. With this interface card in place, my wife and I were able to use our computers just like regular telephones. The computers even rang when someone called us. By this time, I was hooked; VoIP was fun!

It was about this time that my wife and I had grown dissatisfied with our current answering machine, so I configured Asterisk to function in its place. As an answering machine, it worked quite well. My wife and I were able to put messages into folders, forward them to each other, and receive our voice-mail messages as e-mail attachments. As we both read our e-mail regularly, having our voice mail available from our e-mail client was very convenient.

People like to have fun with their answering machine greeting messages, and I'm no different. Because of Asterisk's flexibility, I was able to do something that couldn't be done with a conventional answering machine. My answering machine greeted people by name! When a call came in, the Asterisk system would answer and play a recording of me saying "Hello". Then it would use the caller's caller ID to find a .wav file, 555-1234.wav for example, that contained a recording of me saying the caller's name. Then, finally, it would play our standard greeting. The result was something like "Hello, Tyra Banks, you have reached my super smart answering machine. Please leave a message." Tyra never actually called me, but I'm sure she's just busy.

It also was nice not to have to get up to see who was calling. The Asterisk system was able to send caller-ID information to my MythTV, which displayed who was calling on our television. Sending caller-ID information as a pop-up to my wife's laptop as well as my workstation was pretty simple. But what was really nice was having incoming calls announced over the server's speakers. We actually were able to hear who was calling without having to run over to a phone or caller-ID box. In hindsight, it sounds rather lazy not wanting to be bothered to get to a phone to see who's calling. On the other hand, there are times when we simply don't want to be bothered. Why should I get up from the dinner table to find out who's calling, only to find it's a telemarketer that I don't want to speak with anyway?

The main problem with the Asterisk system at this point was that there was a definite disconnect between the Asterisk system and the "real" phones in the house. For example, we weren't able to use our real phones to check voice mail. The solution to this problem was the addition of an Analog Telephony Adapter (ATA). I chose to buy a Sipura SPA2002. The SPA2002 has one Ethernet port, two telephone ports and speaks the SIP VoIP protocol. The SPA product line is easy to configure with a Web browser. Once the SPA and Asterisk were configured, I could plug two phones in to the system and dial in and out with them.

Of course, I wanted to make the system as transparent as possible, so I had to do something a bit different. I bought a two-line telephone splitter and plugged both "telephone" lines from the SPA into the line one and line two receptacles on the splitter. Then, I plugged the line one and two receptacle into the wall, essentially using the splitter as a joiner. Now I could use any phone jack in the house, either line one or line two. I chose to use line two in the office and line one everywhere else. In most houses, if someone is on the phone, no one else can use the phone. With this setup, if someone is on the phone on line one, we simply go to the office and pick up line two to make our call. This has proven to be very convenient.

The analog telephones have one minor deficiency. The only way to know if you have voice mail waiting for you is to pick up the receiver and listen for the stutter dial tone. Of course, being able to receive voice mail as e-mail attachments is nice, but this situation is only slightly better than having Qwest's voice-mail service. Alas, this minor annoyance didn't improve until I started using standalone IP telephones, which I describe later.

Once I had gotten the home VoIP system working to our satisfaction, I decided to make the big jump by cutting our ties to Qwest and subscribing to a VoIP service instead. Choosing a provider turned out to be more difficult than I had expected. Because I still wanted to use my Asterisk system for call processing, I was able to eliminate several providers right away. Many providers either don't allow you to use Asterisk or don't support the use of Asterisk with their service. With something as complex as VoIP, and as important as my home phone, I really wanted to have support available. As I researched the remaining options, I discovered that many VoIP providers won't allow you to use the service for any commercial calls, including telecommuting and charitable activities. These companies don't publicize this type of policy, but if you read the fine print, you often find that the penalty for getting busted can be steep. At this point, I was down to the wholesale VoIP providers, but this came with the added bonus of being able to get ridiculously low rates, because the provider isn't providing any call features and isn't having to manage a voice-mail system.

Finally, I chose to go with a company called Terravon Communications, although other companies support the use of Asterisk with their service. Now that I was essentially my own phone company, this was when the schooling began. The first thing to note is that most wholesale VoIP providers are on a prepaid basis, so if you forget to pay for next month's service, you get shut off pretty quickly. The symptoms of failing to prepay are difficult to diagnose, because you still have a dial tone, but you can't call anywhere! I also learned, the hard way, that even seemingly innocuous changes to the Asterisk dial plan can cause major problems. I wish I had a dollar for every time I made a late-night tweak to the dial plan only to wake up the next morning to discover that the phones didn't work. Eventually, I learned to test, test, test.

Then there are those rare times that the phones don't work, and you know you didn't change anything. After working through the problem, you decide that the problem must be on the provider's side. So now who are you going to call? Well, nobody, because your phones don't work! Most of your support issues will be handled via e-mail. This is a mixed blessing. On the one hand, it means that you don't get the immediate satisfaction of talking to a warm body. On the other hand, it gives you the opportunity, or obligation, to provide the support staff with all of the information they need to diagnose the problem. I actually have a trouble report template I use on those rare occasions when I encounter problems on the provider's side. I first describe what the problem is. I confirm that I've made no changes to my Asterisk, server or firewall configuration since the last time it worked. I give a specific example of what I am trying to do that isn't working, such as providing a phone number or

EmperorLinux

...where Linux & laptops converge

Portable

Since 1999, EmperorLinux has provided pre-installed Linux laptops to universities, corporations, government labs, and individual Linux enthusiasts. Our laptops range from full-featured ultra-portables to desktop replacements. All systems come with one year of Linux technical support by phone and e-mail, and full manufacturers' warranties apply.

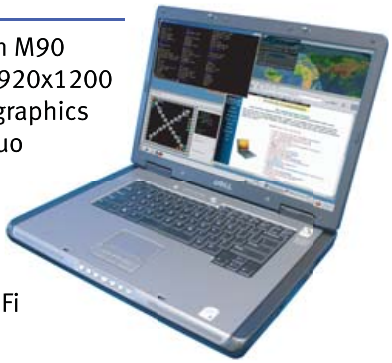
Toucan T60/T60ws

ThinkPad T60/T60ws by Lenovo
- Up to 15.4" WSXGA+ w/ X@1680x1050
- ATI Mobility FireGL V5200
- 1833–2333 MHz Core 2 Duo
- 512 MB–4 GB RAM
- 60–120 GB hard drive
- CDRW/DVD or DVD±RW
- 5.2–6 pounds
- 10/100/1000 Mbps ethernet
- 802.11a/b/g (54Mbps) WiFi
- **Starts at $1950**

Powerful

EmperorLinux specializes in the installation of Linux on a wide range of the finest laptops made by IBM, Lenovo, Dell, Sony, and Panasonic. We customize your choice of Linux distribution to your laptop and provide support for: ethernet, wireless, X-server, ACPI power management, USB, EVDO, PCMCIA, FireWire, CD/DVD/CDRW, sound, and more.

Rhino D820/M90



Dell Latitude D820/Precision M90
- Up to 17" WUXGA w/ X@1920x1200
- NVidia Quadro FX 3500M graphics
- 1667–2333 MHz Core 2 Duo
- 512 MB–4 GB RAM
- 40–160 GB hard drive
- CDRW/DVD or DVD±RW
- 6.3–8.6 pounds
- 802.11a/b/g (54Mbps) WiFi
- ExpressCard/EVDO
- **Starts at $1455**

Unique

EmperorLinux offers Linux laptops with unique features. Ruggedized Panasonic laptops are designed for harsh environments: drops, vibrations, sand, rain, and other extremes. ThinkPad tablet PCs are like other laptops, with an LCD digitizer for pen-based input both as a mouse and with pressure sensitivity for writing and drawing on-screen.

Raven X60 Tablet

ThinkPad X60 Tablet by Lenovo
- 12.1" SXGA+ w/ X@1400x1050
- 1667–1833 MHz Core Duo
- 1–4 GB RAM
- 80–120 GB hard drive
- 4 pounds
- Pen/stylus input to screen
- Dynamic screen rotation
- Handwriting recognition
- X60s laptops available
- **Starts at $2300**

www.EmperorLinux.com 1-888-651-6686

area code that I can't reach. Finally, I give them the relevant timestamped log entries from /var/log/asterisk/messages. I usually start by truncating the log file and telling Asterisk to start logging either IAX or SIP debug messages. Then, I do whatever it takes to repeat the symptoms. All of the messages that the servers exchanged, as well as the steps that my Asterisk server executed are now in the log file, which I send as an attachment to the tech support staff. You can save a lot of time by making sure that the engineers have all of the information they need—the first time.

One day, I got a call from a coworker who had heard that I had my own VoIP server. One of his friends was going to be in Europe for about three weeks and they wanted to know if I could set him up. He had a wireless PDA that ran Windows CE and wanted to know if he could connect it to my server and use it like a phone to call his friends and family in the United States, from Italy. Because this sounded interesting, I agreed to give it a try. He installed a copy of SJphone, and I configured it to talk to my Asterisk server. Then he went to Europe. Amazingly, it worked pretty well! I talked to him for several minutes, and it sounded fine, though I was a bit jealous to hear that he was sitting in a wireless cafe in Venice, Italy, while I was sitting in my office. The whole experiment cost me only $6 US in line charges, so it was worth it just for the nerdiness of it. I shudder to think what it would have cost him to call home from Europe with a land line, much less what it would cost if he had tried to use his cell phone.

I was already able to use any phone or computer in the house to make and receive phone calls, but I wasn't done exploring. I managed to borrow a Cisco 7960 IP telephone. This was a substantial, though temporary, improvement to our telephone system. The 7960 is a very attractive, business-looking phone. It also features a sharp LCD display that it uses to display context-sensitive menus as well as caller-ID information. But most important for me, it has a bright-red message-waiting indicator that we can see from across the house. However, at $300 US, the 7960 isn't in my budget.

Because the 7960 I had borrowed didn't allow Web-based configuration, it was a little bit more difficult to configure than the SPA2002. I had to create a configuration file that the 7960 downloaded from a TFTP server, which I also had to install. Fortunately, example configuration files are available on the Internet.

Finally, I decided to buy my own IP telephone and chose the Polycom Soundstream IP501, which I bought at VoipSupply.com for about $180 US. This positions the IP501 as a midrange telephone, suitable for either the office or home. The IP501 is a nice-looking device, though perhaps not as attractive as the Cisco device. It also has a red, blinking, message-waiting light. The LCD display seems smaller, but it's functional. The big plus with the IP501 is the way it sounds. Going from the Cisco phone to the Polycom phone is like going from AM radio to FM; it just sounds better.

The IP501 has a rudimentary Web-based configuration capability, though the real meat of the device's capability is exposed only via a configuration file that the device downloads via FTP or HTTP. Fortunately, the Polycom Web site contains complete examples of these configuration files as well as a 166-page Administrator's Guide.

Both the Cisco and Polycom phones feature a two-port switch that allows you to plug your PC workstation in to the phone, and then plug the phone in to the network. In this configuration, the phone will prioritize the real-time voice traffic over the rest of the network traffic, thus ensuring that voice quality is as good as it can be. Also, both devices are capable of negotiating separate VLANs for both the phone and the attached PC. This allows your voice traffic to travel over a separate, perhaps more secure, network. The network infrastructure has to support VLAN negotiation for this to work.

In the year or so that I've been using Asterisk, I've learned a lot and had a lot of fun. Starting from a standalone "toy" system, I added PC-based softphones. Later, the system was connected to the PSTN. Then, the system was connected to the house phone wiring. Finally, I added dedicated IP telephones. I ended up with a phone system that rivals those of many large offices. Each configuration, as well as each device, has its benefits as well as drawbacks. They all have their own unique quirks. Learning about these quirks is what makes VoIP so much fun.

—MIKE DIEHL

Are you shocked by the high cost of iSCSI & Fibre Channel storage?

AoE is your answer!

ATA-over-Ethernet = simple, low cost, expandable storage.

www.coraid.com

Winner
Product Excellence Award
LinuxWorld Conference & Expo
Best Storage Solution

EtherDrive® SR1520

- RAID enabled 3U appliance with 15 slots for hot swap SATA disks
- Check out our other Storage Appliances and NAS Gateway

1. Ethernet Storage – without the TCP/IP overhead!
2. Unlimited expandability, at the lowest possible price point!!
3. You want more storage…you just buy more disks – it's that simple!!!

Visit us at www.coraid.com for more information.

CORAID

1.706.548.7200 The Linux Storage People **www.coraid.com**

REUVEN M. LERNER

Dojo

Become a black belt in JavaScript in your very own Dojo.

JavaScript has experienced a renaissance in the past year or two. Whereas many Web developers long saw JavaScript as a second-class programming language, useful for (at best) decorating Web pages, it is an increasingly central technology for Web developers. Whether you are adding Ajax (Asynchronous JavaScript and XML), dynamic HTML or new GUI widgets to your Web pages, you likely have begun to use JavaScript more in the past year or two than ever before.

Luckily for all Web developers, the rapid and widespread interest in JavaScript programming has resulted in the development of JavaScript libraries and toolkits, many released under an open-source license. In my last few columns, we looked at Prototype, which aims to make general JavaScript programming easier, and at Scriptaculous, which provides visual effects and interface widgets. Prototype has become quite popular among open-source programmers, in no small part because of its inclusion in the Ruby on Rails application framework.

But, Prototype and Scriptaculous are far from the only games in town. Another popular open-source JavaScript framework is Dojo. Dojo is based on a number of convenience classes and objects begun by Alex Russell of JotSpot, a startup purchased by Google in 2006. Russell continues to work on Dojo, but contributions of code and money now come from other sources as well, including companies such as Sun and IBM. Moreover, Dojo is now included by default in the popular Django Web application framework, giving it additional exposure.

This month, we take an initial look at Dojo, examining the way it divides code into packages, then at several of the convenience functions it provides for JavaScript programmers and, finally, at a very small sample of Dojo's large widget library. Even if you have no intention of using Dojo, I hope you find this article instructive. I almost always find it useful to see how other languages and toolkits do things, if only to get some better perspective on what I am doing.

Installing Dojo

The first thing to understand about Dojo is that it is large, at least by JavaScript standards. (Remember that all JavaScript code must be downloaded from the server, interpreted by the browser and then executed within memory, all as quickly as possible. A large JavaScript library might offer many features, but it will make performance unacceptably slow.) Thus, although we might consider Dojo to be a single, large library, it is actually a collection of many smaller parts.

This is a relevant point even before you download Dojo, because the download site requires that you choose which combination of features you prefer to use. Knowing that my server is on a relatively high-speed line, that my sites tend to be relatively lightweight and that I plan to explore Dojo as a developer, I installed the "everything" version, labeled as kitchen sink on the download site. But, if you are interested in Dojo solely for its rich-text editor, or for use in Ajax or charting, you might want to download one of the many smaller versions, each identified by the subset of Dojo's functionality it covers. For the purposes of this column, however, I assume you also have downloaded the kitchen sink version.

At the time of this writing, Dojo is at version 0.4.1, and the kitchen sink version is available from the URL **download.dojotoolkit.org/release-0.4.1/dojo-0.4.1-kitchen_sink.tar.gz**.

Once you have downloaded the file, unpack it:

```
tar -zxvf dojo-0.4.1-kitchen_sink.tar.gz
```

The directory that you open will contain a number of different items, including a README file, several Flash animation (*.swf) files used in Dojo's persistent storage engine, the main dojo.js JavaScript file and several subdirectories, including one containing demos (called demos), one containing the source code for much of Dojo's functionality, and release and tests directories for development of the toolkit itself.

To get Dojo up and running on your server, you must put dojo.js and the src subdirectory under your document root. I tend to put my Web sites under /var/www/SITENAME/www, and JavaScript files go in the javascript directory under that path. I created a further subdirectory named dojo and put both dojo.js and the src directory there as well. Thus, the full path to dojo.js on my filesystem is /var/www/SITENAME/www/javascript/dojo/dojo.js, but the URL that we will use to load it from a Web page will be /javascript/dojo/dojo.js.

And, indeed, we can load Dojo into our Web pages using the standard <script> tag:

```
<script type="text/javascript"
src="/javascript/dojo.js"></script>
```

Although the above loads dojo.js into the browser's memory, this does not mean all of Dojo's commands are now available. Rather, including dojo.js makes it possible for us to load one or more of Dojo's individual packages. You can think of dojo.js as a bootloader, in that its only purpose is to make Dojo available to you later on, rather than to perform any tasks on its own.

Dojo Packages

As we saw during the past few months, Prototype and Scriptaculous have fairly well-defined roles, and thus, they remain separate products. Prototype provides a large number of convenience functions for JavaScript programmers,

and Scriptaculous adds GUI-related functionality on top of it. Dojo is designed with a different organizational philosophy in mind, providing a wide array of different functions, many of which might seem unrelated to one another.

For example, Dojo provides GUI elements (for example, a rich-text editor, a date picker, interfaces to mapping sites and layout containers). But, it also provides an event system, making it possible to assign functionality to particular events, using a variety of different models. It provides a client-side storage system with more sophistication than HTTP cookies. It provides a number of utilities for JavaScript programmers, making it possible to create new classes, send notes to a debugger or otherwise work with the language.

Each of these pieces of functionality is available inside of a separate package, which is both loaded and identified with a hierarchical name structure. Thus, all Dojo functions begin with dojo (for example, dojo.declare and dojo.debug), and they are loaded as part of a similarly named hierarchy.

Loading a Dojo package is as simple as putting:

```
<script type="text/javascript">
dojo.require("dojo.PACKAGE.*");
</script>
```

inside your HTML. You can load more than one package, using multiple invocations of dojo.require. Dojo's package loader is smart enough to take care of any dependencies that might exist.

JavaScript Helpers

Once you have included Dojo, you can begin to use some of its improvements to the JavaScript language. Dojo includes a number of convenience functions to make JavaScript programming easier, some of which are quite similar to what Prototype offers. For example, nearly every JavaScript program needs to retrieve nodes based on their id attributes. (An id attribute is supposed to be unique in a particular page of HTML, thus allowing us to identify a node uniquely.) To assign the variable myNode to the node with the ID of target, we normally would need to write:

```
var myNode = document.getElementById("target");
```

Dojo allows us to abbreviate this to:

```
var myNode = dojo.byId("target");
```

This is not quite as short as Prototype's $() operator, but it is still a significant improvement, making programs both shorter and more readable.

Dojo also provides some new mechanisms to work with arrays and other enumerated lists. For example, it provides a foreach loop:

```
dojo.lang.forEach(arrayName, iterationFunctionName);
```

The above code causes iterationFunctionName to be invoked once for each element of arrayName. Thus, we could say:

```
var names = ["Atara", "Shikma", "Amotz"];
dojo.lang.forEach(names, alert);
```

to print each of these names in an alert box. Dojo provides several other convenient functions for use with arrays, including dojo.map (which invokes an operation on each element of an array, producing a new array as a result) and dojo.filter (which returns an array of those items for which a function returns true). Stylistically, the documentation for Prototype seems to encourage users to write inline functions, whereas the Dojo documentation encourages users to write named functions and then refer to them. However, you can adopt whichever style is more appealing to you.

Rich Text

One of the easiest parts of Dojo to begin using is its collection of widgets. From the time that HTML forms were first standardized, Web developers have wanted a richer set of widgets from which to choose, in order to provide applications that resemble—in style, as well as power—parallel widgets available for desktop applications. Dojo provides a number of such widgets, making it possible to include rich-text editors, sliders and combo boxes in our programs.

For example, we might want to use the Dojo rich-text editor, allowing people to write using more than the plain text that a <textarea> tag provides. We can do that simply by creating a and giving it a class of dojo-Editor:

```
<div class="dojo-Editor">
    Hello from the Dojo editor!
</div>
```

If you fire up the above, you'll get...nothing, other than a with some text inside of it, as you might expect without installing Dojo. This is because of Dojo's modular loading scheme; loading dojo.js is only the first step in using any of Dojo's functionality, bootstrapping the loading system. Loading the actual editor code requires that we invoke the dojo.require JavaScript function:

```
<script type="text/javascript">
dojo.require("dojo.widget.Editor");
</script>
```

Once we have done this (producing the file shown in Listing 1), we suddenly have a rich-text editor at our disposal. This is wonderful, except for one thing—how do we submit the HTML-formatted file to an application on our server? One method would be to use Ajax to save the contents of our div on a regular basis, submitting its contents to the server without any need for explicit saving. And, indeed, this is what many Web-based applications, including word processors and spreadsheets, have done. No longer do you need to

The first thing to understand about Dojo is that it is large, at least by JavaScript standards.

Listing 1. simple.html

```
<html>
    <head>
        <title>Dojo page title</title>
        <script type="text/javascript"
        src="/javascript/dojo.js"></script>

        <script type="text/javascript">
            dojo.require("dojo.widget.Editor");
        </script>
    </head>

    <body>
          <div class="dojo-Editor">
              editor goes here
          </div>
    </body>
</html>
```

Listing 2. simple-form.html

```
<html>
    <head>
        <title>Dojo page title</title>
        <script type="text/javascript"
        src="/javascript/dojo.js"></script>

        <script type="text/javascript">
            dojo.require("dojo.widget.Editor");
        </script>
    </head>

    <body>
       <form method="POST" action="/parse.php">
           <textarea name="text" class="dojo-Editor">
               write something!
           </textarea>
         <input type="submit" />
       </form>
    </body>
</html>
```

save documents; you simply work with them, and you can expect the computer to save what you've done reliably.

This month, we take a simpler approach, including our rich-text editor in an HTML form submission. Unfortunately, it's still not obvious how we can pull this off, because HTML forms consist of <input> and <textarea> tags. Luckily, the Dojo team has taken this into consideration. Notice that we define a Dojo widget by its class, not by its tag type. This means we can attach the dojo-Editor class to anything, not only to an empty tag. If we attach it to a <textarea>—which is a block element, just like —our text editor will be attached to a textarea, which will be submitted to the server. In other words, we replace our with:

```
<textarea name="text" class="dojo-Editor">
    Hello from the Dojo editor!
</textarea>
```

Listing 2 shows an example of how this might look when incorporated into a simple HTML form. When the contents of the form are sent to the server, all formatting is preserved using HTML tags. Your application will need to parse this HTML to understand any formatting that might appear in the text.

Of course, if your plan is to take the input text and simply display it in a Web browser, not much (if any) work is needed on your part. You can stick the input into a database and then retrieve it whenever it is needed. (I haven't checked into the security of this widget to make sure it is immune to cross-site scripting attacks, so you might want to investigate it further before simply accepting, storing and displaying user data.)

Conclusion

As you can already see, Dojo offers a wide variety of functions and doesn't take much effort to start using. But using many of the other widgets Dojo includes, such as an attractive DatePicker, requires that we use Dojo's sophisticated event handler, which we did not examine here. Next month, we will look at events in Dojo and how that package lets us incorporate special effects, Ajax and many more widgets into our Web applications.■

Reuven M. Lerner, a longtime Web/database consultant, is a PhD candidate in Learning Sciences at Northwestern University in Evanston, Illinois. He currently lives with his wife and three children in Skokie, Illinois. You can read his Weblog at altneuland.lerner.co.il.

Resources

The main source for information about Dojo, as well as Dojo software releases, is at **dojotoolkit.org**. Documentation for the toolkit is still a bit sparse, but it has improved significantly in the last few months, and continued improvements seem likely, given Dojo's growing popularity. The main URL for Dojo documentation is at **dojotoolkit.org/docs**, with Dojo.book (the Wiki-based Dojo documentation) at **manual.dojotoolkit.org/index.html**.

Some good articles about JavaScript toolkits, including Dojo, are at **www.sitepoint.com/article/javascript-library**.

Finally, a good introduction to rich-text editing with Dojo is at **dojotoolkit.org/docs/rich_text.html**.

Free Long Distance—Really!

MARCEL GAGNÉ

If you are still recovering from the last time you paid your long-distance bill, Chef Marcel may be able to offer some relief.

Yes, *mes amis*, there really is an Ubuntu wine, distributed by the South African winemaker KWV.

Mon Dieu! François, look at this phone bill! Explain these long-distance charges. Yes, I know you have friends and family all over the world, but why are you calling them from the restaurant telephone? *Quoi?* Because it is too expensive to use your own phone? François, aside from the fact that you have used up two months of your salary with these calls, there are better ways to save money on long distance. How? By using your Linux system and a Voice over IP program, of course. That just happens to be the focus of tonight's menu.

We will discuss this bill later, *mon ami*. I can already see our guests starting to arrive. Head to the wine cellar, *immédiatement*. Bring up the 2004 South African Ubuntu Shiraz we were sampling earlier today. *Vite!* Perhaps if you are truly efficient, I may re-instate your salary despite these calls. Enough smiling. *Vite!*

Ah, welcome, *mes amis*, to *Chez Marcel*, home of fine wine and fine Linux fare. Please sit and make yourselves comfortable. I have sent François to the wine cellar, and he should be back with your wine very shortly. While we wait, however, let me ask a question. Who, besides myself, would like to avoid paying hefty long-distance fees? I see. Well, how does free long distance to anyone anywhere in the world sound? That's what I thought.

Figure 1. If you don't have a SIP address yet, Ekiga makes it easy. Click the link provided, and you'll be set up in no time.

VoIP (Voice over IP) programs allow you to communicate with others running similar programs on their respective computers. All you need, aside from a computer, is a microphone and speakers, though it's often a good idea to use a headset that combines these two. Your voice is transmitted via packets over your existing Internet connection to a receiving system, wherever it might be.

Just as with cell phones that include every feature imaginable, from cameras to MP3 players, it's getting harder and harder to find a program that's just a phone. Many offer advanced features, such as call answer, conference calling and a whole lot more. Using the right program, you even can transmit video, just like they did back in the 1960s on *2001: A Space Odyssey*. To transmit video, you also need a Webcam. These inexpensive USB cameras are used for a variety of things, providing glimpses into the life of the individual running a particular Web site. Other sites provide a camera to reassure us that they are indeed working. Others still are there to let parents observe their children playing at day care. As it turns out, they also help make great video phones.

Ah, François, you have returned. Please, pour for our guests. Make sure everyone's glass is full. There are a couple of things we should talk about before we look at the phones on today's list.

Where shall we start? *Mais oui*—protocols, protocols, protocols. SIP, or Session Initiation Protocol, is (as the name implies) a protocol used to create, manage and end sessions between two or more users. The sessions we are talking about in this case are phone conversations. All of the programs on tonight's menu are SIP phones. As you know, here at *Chez Marcel*, we love open source and we love open protocols. A wonderful thing about all these programs is that every one of them can call another's SIP address. The second thing we need to cover is security and some of the problems it poses to VoIP programs.

If you are running these packages on your corporate or home LAN, you should have no problems. The same holds true if you are running them from a single machine connected to the Internet—odds are this will work without a hitch. The catch comes when you try to work from behind a masqueraded (or NATed) firewall. Luckily, many of these programs use STUN to help with this little problem.

STUN stands for Simple Traversal of UDP through NATs (which stands for Network Address Translations). Aren't you glad they just call it STUN? The whole point is to provide a

protocol that helps systems working from behind a NAT firewall route their packets to and from the appropriate place. To use STUN, your program must register with a STUN server, a number of which are public and free to use.

That should take care of the background information. Let's take a look at some SIP phones now, starting with a little something called Ekiga. Once upon a time, there was a program called GnomeMeeting. These days, it's called Ekiga, and it is a great VoIP program. Ekiga, which supports both SIP and H323 protocols, is an excellent telephone application as well as a great video conferencing tool. It even will work with Microsoft NetMeeting. You'll find that Ekiga is rich in features with multiple, simultaneous account support, call hold, call transfer, call forwarding, instant messaging and a whole lot more.

Ekiga is available from its Web site (see Resources), where you'll find binaries for a number of popular distributions (and source, of course). Ekiga also has a .net sister site from which you can get your own SIP address so that people can call you or look you up in the on-line directory. In fact, when starting Ekiga for the first time, you are presented with the the First Time Configuration Assistant (Figure 1). Part of this process involves registering a free Ekiga.net SIP account (if you don't already have one). You can enter your address here, or click the link provided to set one up.

When you are happy with the information you are presenting, click the Forward button to continue, and you will be asked to specify the type and speed of connection you are using. When you click Forward past this screen, you'll come face to face with the issue of security I mentioned earlier. Click the box labeled Detect NAT Type, and Ekiga's Assistant will try to determine what kind of routing assistance you may need. If the program determines that you are behind a NAT firewall, you'll be given the opportunity to enable STUN support.

Click Forward. For the next few screens, the Assistant will help you configure your sound card, microphone and Webcam. As you go through the Configuration Assistant, buttons are provided to test your chosen settings, whether they be audio or video. By the way, Ekiga also provides support for FireWire digital cameras (like the digital video camera you bought to take videos of your new baby) through pwlib plugins. If you are going to use one of these cameras, make sure you also download and install the pwlib-plugins-avc package. You always can configure a different camera later by clicking Edit on Ekiga's menu bar, selecting Preferences, and then looking under Video Devices. Change the Video Plugin over on the right (Figure 2), and your camera should be detected.

Of course, you don't need a Webcam (or any kind of camera) if you just plan to use Ekiga as a SIP phone. Click Apply on the final screen, and Ekiga fires up. You can configure several options with the package through the preferences menu. You also can re-run the First Time Assistant at any time.

To place a SIP call, type `sip:friendsaddress@ekiga.net` in the address bar just below the menu bar—this example assumes that your friend's SIP address is a registered Ekiga.net address (Figure 3). A little pop-up window appears on the second PC warning the user of an incoming call. If you accept the connection, the two clients can communicate.

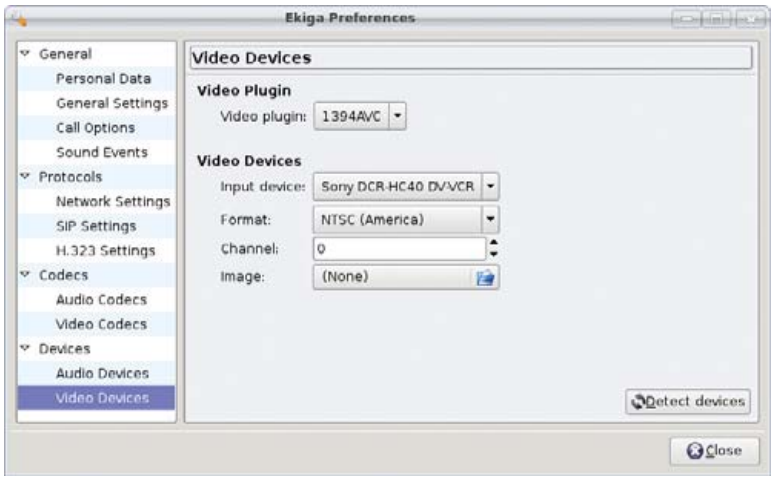


Figure 2. With the right plugin, Ekiga can configure and use a FireWire digital camera.

Notice the button bar to the left in Figure 3. You can turn video or audio on and off, and you can bring up a chat window for text message exchange. In fact, Ekiga will work under a number of different configurations. You can run video only, audio only, text only or any combination of the three modes. If you have a camera and your friends don't, that's okay too, because they'll still be able to see you. Of course, it can be a little disconcerting to know that

Figure 3. I wonder if Mom and Dad are available for an Ekiga chat?

Figure 4. WengoPhone aims to be a complete messaging solution.

Figure 5. If you don't already have an account, this is your opportunity.

somebody out there can see you, but you can't see them.

When you run Ekiga, make sure you turn on the control panel. It opens up to a tabbed window in the application, providing support for audio and video controls as well as a dialpad. This shows the status of calls, your registration with on-line directories and other information, and it can be turned on or off at any time without affecting the transmission. Ekiga also provides an address book, so you can search users who are registered with Ekiga.net. Simply click the address book icon on the left icon bar.

François, our guests' glasses are looking a little dry. Please offer everyone a refill while I present the next item on tonight's menu.

WengoPhone is a fantastic VoIP client, and one you must check out. First and foremost, WengoPhone has great sound quality, a must for a VoIP application. It features free Wengo-to-Wengo communications worldwide; inexpensive Wengo-to-standard-telephone calling (using Wengo credits); conference calling; SMS support; instant messaging to Yahoo, MSN, AIM/ICQ and Jabber (such as Google Talk) clients; as well as video chat. Wengo is a very slick and feature-rich application. I'll tell you all about it, but start by visiting WengoPhone's Web site and downloading a copy.

When WengoPhone starts, it asks you to log in using the e-mail address with which you registered and your password (Figure 5). If this is your first time, you likely won't already have a Wengo account, so click the Click here if you don't have a Wengo account button. A browser window appears on the Wengo Web site where you can register your account. On doing so, you'll get a small Wengo credit you can use to call land lines and cell phones (to which you also can send text messages).

After you are registered, Wengo starts up, finding and configuring your audio and video automatically. This isn't to say it will find everything you have, but it tries. Down at the bottom right-hand corner of the main WengoPhone window (Figure 4), there are little icons telling you the status of your connection, registration, audio and other things. If there's a problem with the audio, for instance, an icon appears with a tooltip alerting you of the trouble. You can then click the icon to make adjustments to the levels or to visit the configuration screen.

You can fine-tune a number of WengoPhone's features through the configuration dialog. Click Tools on the menu bar, and select Configuration to open the settings window. In Figure 6, I am using the dialog to enable and configure my Webcam.

I mentioned previously that WengoPhone supports instant messaging. This is a great way to clean up some application clutter on your desktop. If you're using one or more instant messaging clients in addition to a VoIP application, Wengo can help you out. To use WengoPhone as your instant messenger, click Accounts in the configuration window's sidebar. Then, select Add and enter your screen names and passwords for either Jabber/Google Talk, MSN, AIM/ICQ or Yahoo. The configuration dialog also lets you set a call-forwarding address, sounds, language or auto-away timeouts.

THE PENGUIN

Another server down. Another night at the office. Whether it's deploying a hundred new servers or installing the latest security patch, it doesn't matter. You're sleeping with the servers again. Penguin Computing® introduces Scyld *Webmaster™ Infrastructure Manager*. Its centrally-managed, highly available architecture makes large pools of Linux servers act like a single, consistent virtual system, significantly reducing complexity and time to deploy servers on the fly. Its highly secure environment makes your server farm a 'virtual fortress' and its simplified manageability cuts your maintenance windows and administration by up to 80%, dramatically improving TCO. So go on home, catch some zzzzs and know that Penguin is standing guard.

MAY CAUSE DROWSINESS

Highly SCYLD

AMD 64 Opteron

PENGUIN HIGH DENSITY CLUSTER. The highest density, modular blade server architecture on the market. With powerful Scyld *WebMaster™ Infrastructure Manager* for single point command and control, and AMD Dual Core Opteron™ for a highly productive user experience.

www.penguincomputing.com

PENGUIN COMPUTING

REALLIFELINUX

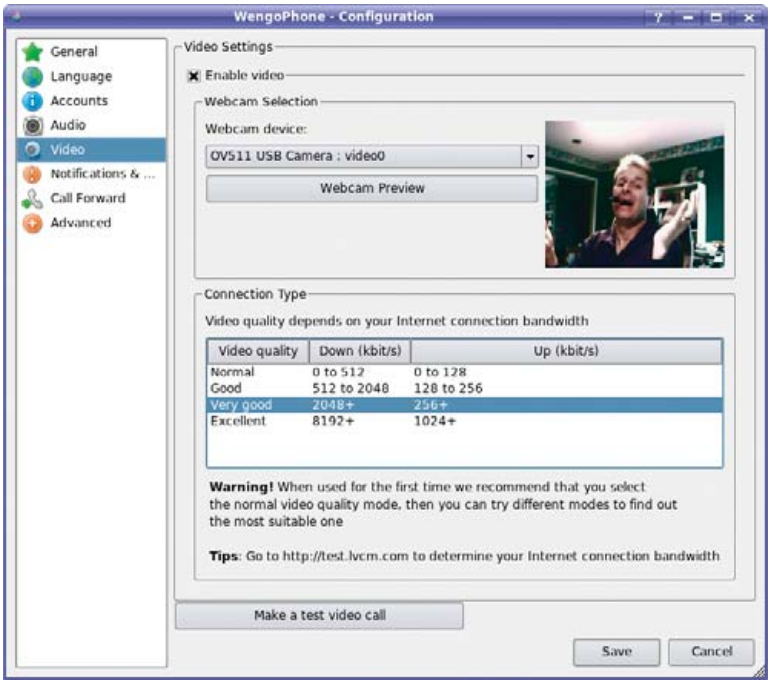


Figure 6. Use WengoPhone's configuration dialog to enable video support.

To call someone with your WengoPhone, enter a SIP address into the location bar at the bottom (Figure 4, again). Click the green call button, and your party's SIP phone rings. This could just as easily be a Wengo address as an Ekiga address. To call a standard telephone, enter a plus sign followed by the complete number, including country code (where applicable) and area code.

Using WengoPhone, you can send text messages to cell-phone users with the SMS feature. Simply click the SMS icon, and a small composer window appears (Figure 7). Enter your contact's phone number (remember the plus-sign number format) or select it from your address book, then enter your message. A counter lets you know how many characters you have used and how many are available. When you are done, click Send. For people, like your Chef, who avoid text messages because they don't like thumbing, this is a great little feature.

SMS, like calling regular phones of other non-Wengo users, does cost credits, which you can buy directly from your WengoPhone, but the price is very reasonable. In fact, calling the rates inexpensive seems unfair—downright cheap sounds more appropriate. As I write this, a call from my PC in Ontario, Canada, to England is .008 cents (Euro) per minute. Yes, those are two zeros to the right of the decimal point.

Directly to the right of your nickname on the status bar, you'll see a counter indicating the amount left in your Wengo credits account. Click the total, and an information window expands below, with a small menu of options. One of these is labeled Purchase call out credits. Click here, and a browser window appears from which you can make your purchase or discover rates.

I fear, *mes amis*, that closing time is already upon us, with plenty of other VoIP phones left uncovered. Gizmo offers another great softphone rich in features. Although the Linux client did not yet support video at the time of this writing, Gizmo does allow you to record calls (with great sound quality). It also has some amusing features, such as configurable hold music and audio sound effects (for example, rolling thunder or a tiger's roar) that you could insert into your calls. Although it's not as feature-rich as some of the softphones covered on today's menu, you also might want to check out KPhone or Linphone. I'll leave you to explore those when you return home, but the clock on the wall is truly insistent.

Figure 7. Send text messages to your friends without getting your thumbs in a knot.

François, kindly refill everyone's glass one more time so that we may raise a toast. And now, *mes amis*, raise your glasses and let us all drink to one another's health. *A votre santé! Bon appétit!*■

Marcel Gagné is an award-winning writer living in Waterloo, Ontario. He is the author of the all-new *Moving to Free Software*, his sixth book from Addison-Wesley. He also makes regular television appearances as Call for Help's Linux guy. Marcel is also a pilot, a past Top-40 disc jockey, writes science fiction and fantasy, and folds a mean Origami T-Rex. He can be reached via e-mail at mggagne@salmar.com. You can discover lots of other things (including great Wine links) from his Web site at www.marcelgagne.com.

Resources

Ekiga: **www.ekiga.org**

Ekiga's Sister Site: **www.ekiga.net**

Gizmo Project: **www.gizmoproject.com**

KPhone: **sourceforge.net/projects/kphone**

Linphone: **www.linphone.org**

WengoPhone: **www.wengophone.com**

Marcel's Web Site: **www.marcelgagne.com**

The WFTL-LUG, Marcel's On-line Linux User Group: **www.marcelgagne.com/wftllugform.html**

servers DIRECT™

MORE PRODUCTS, BETTER SERVICE, GUARANTEED.

GO STRAIGHT TO THE SOURCE!

MANAGE RISKS, REDUCE COSTS.

THE DUAL-CORE INTEL® XEON® PROCESSOR IN YOUR SERVERSDIRECT SYSTEM OFFERS BUILT-IN TECHNOLOGIES TO ENCHANCE SECURITY AND RELIABILITY

SDR-3500T 3U 64-BIT XEON DATABASE SERVER

Designed to provide outstanding performance with dual-core Xeon® 5000/5100, ideal solution for storage and mission critical business application server

FEATURE HIGHLIGHTS:

- Intel Xeon 5050 3.0GHZ Dual Core Processor (Dual Processor Option)
- 3U Chassis with 800W Redundant Power Supply
- Intel® 5000P (Blackford) Chipset
- Kingston 1024MB 667MHz DDR2 ECC FB-DIMM(2pcs x 512MB)
- Seagate 400GB SATA-II 16MB Cache 7200RPM
- 16 x 1" Hot-swap SATA Drive Bays
- ATI ES1000 Graphics with 16MB video memory
- Intel® (ESB2/GIlgal) 82563EB Dual-port Gigabit Ethernet Controller
- RAID 0, 1, 5, 10 Support

STARTING PRICE **$2,499**

1U ENTRY LEVEL SERVER

1U Pentium® D servers support the latest dual-core architecture that delivers unmatched multi-tasking capabilities in an uni-processor environment

- 1U Rackmount Chassis with 300W Power Supply
- Intel® Pentium® D 930 3.0GHZ Dual Core 800FSB
- Intel® E7230 (Mukilteo) Chipset
- Kingston 1024MB 667MHz DDR2 ECC (2pcs x 512MB)
- Seagate 250GB SATA-II 7200RPM hard drive
- 2 x 1" Hot-swap SATA Drive Bays
- ATI RageXL graphics, 8MB
- 2x Intel® 82573V (Tekoa) PCI-e Gigabit LAN Ports
- RAID 0, 1

$999 SKU# SDR-1110T

COST EFFECTIVE 2U SERVER

Businesses with 50 or more employees seeking maximum performance, storage room, expandability and uptime

- Intel Xeon 5050 3.0GHZ Dual Core Processor (Dual Processor Option)
- 2U Chassis with 600W Power Supply
- Intel® 5000V (Blackford VS) Chipset
- Kingston 1024MB 667MHz DDR2 ECC FB-DIMM (2pcs x 512MB)
- Seagate 400GB SATA-II 16MB Cache 7200RPM
- 6 x 1" Hot-swap SATA-II Drive Bays
- ATI ES1000 Graphics with 16MB video memory
- Intel® (ESB2/Gilgal) 82563EB Dual-port Gigabit Ethernet Controller
- RAID 0, 1, 5, 10 support

$1,599 SKU# SDR-2501T

4U TOWER/RACKMOUNT 64-BIT XEON ENTERPRISE SERVER

Powered by the latest dual-core Xeon 5000/5100® sequence processors, versatility in either a tower or rackmount 4U form factor and support enterprise-class

- Intel Xeon 5050 3.0GHZ Dual Core Processor (Dual Processor Option)
- 4U Rackmountable / Tower
- Intel® 5000X (Greencreek) Chipset
- Kingston 1024MB 667MHz DDR2 ECC FB-DIMM (2pcs x 512MB)
- 8pcs x Seagate 400GB SATA-II 16MB Cache 7200RPM
- 3Ware 9550SX-8port RAID Controller Card
- GeForce 6200 TC 128MB OB 256M support DDRTV-out PCI-E
- Intel® (ESB2/Gilgal) 82563 Dual-port Gigabit Ethernet Controller

$3,999 SKU# SDP-7045A-TB

SDR-5500T 5U ADVANCED STORAGE SERVER

Powered by the latest dual-core Xeon 5000/5100® sequence processors, the absolute best storage capacity available in a 5U format

- Intel Xeon 5050 3.0GHZ Dual Core Processor (Dual Processor Option)
- 5U Chassis with 24 hot-swap bays & 950W redundant power supply
- Intel® 5000P (Blackford) Chipset
- Kingston 1024MB 667MHz DDR2 ECC FB-DIMM (2pcs x 512MB)
- 12pcs x Western Digital 400GB SATA RAID Drive
- 1pc x 3Ware 9550SX-12 port RAID Controller Card
- ATI ES1000 Graphics with 16MB video memory
- Intel® (ESB2/Gilgal) 82563EB Dual-port Gigabit Ethernet Controller
- RAID 0, 1, 5, 10 support

$6,799 SKU# SDR-5500T

SERVERS DIRECT CAN HELP YOU CONFIGURE YOUR NEXT HIGH PERFORMANCE SERVER SYSTEM - CALL US TODAY!

Our flexible on-line products configurator allows you to source a custom solution, or call and our product experts are standing by to help you assemble systems that require a little extra. Servers Direct - your direct source for scalable, cost effective server solutions.

intel Xeon inside Dual-core. Do more.

1.877.727.7887 | www.serversdirect.com

DAVE TAYLOR

Compact Code and Cron Contraptions

It's a simple job to do a cron job.

This month, I thought I'd take another sidetrack. (You knew that entrepreneurs all have ADD, right?) So, it should be no surprise that to me, the fastest way from point A to point B is, um, what were we talking about?

Maximum Capability, Minimum Code

Reader Peter Anderson sent in a code snippet that offers up a considerably shorter way to convert a really big byte count into kilobytes, megabytes and gigabytes than the one I shared in my December 2006 column.

His question: "Why so much extra code?"

His snippet of code to do this takes advantage of the built-in math capabilities of the Bash shell:

```
value=$1
((kilo=value/1024))
((mega=kilo/1024))
((giga=mega/1024))
echo $value bytes = $kilo Kb, $mega Mb and $giga Gb
```

Peter, you're right. This is a succinct way of solving this problem, and it's clear that a shell function to convert, say, bytes into megabytes easily can be produced as a one-liner. Thanks!

As I've said in the past, I don't always write the most concise code in the world, but my goal with this column is to write maintainable code and to get that prototype out the door and be ready to go to the next thing as fast as possible. That practice isn't always compatible with the quest for elegance and perfection in the coding world, to say the least!

Coding with Crontab

On an admin mailing list, I bumped into an interesting question that makes for a perfect second part to this column—a simple script that's really just a one-line invocation, but because it involves the cron facility, becomes worth our time.

The question: "I need to run a cron job that looks in a certain directory at the top of every hour and deletes any file that is more than one hour old."

Generally, this is a job for the powerful find command, and on first glance, it can be solved simply by using an hourly cron invocation of the correct find command.

For neophyte admins, however, there are two huge steps involved that can be overwhelming: figuring out how to add a new cron job and figuring out the correct predicates for find to accomplish what they seek.

Let's start with find. A good place to learn more about find, of course, is the man page (`man find`), wherein you'll see there are three timestamps that find can examine. ctime is the last changed time, mtime is the last modified time and atime is the last accessed time. None of them, however, are creation time, so if a file was created 90 minutes ago but touched or changed eight minutes ago, all three will report eight minutes, not 90. That's probably not a huge problem, but it's worth realizing as a very typical compromise required to get this admin script working properly.

For the sake of simplicity, I'll actually change this example to deleting files that haven't been accessed in the last 60 minutes, not worrying about how much earlier they might have been created. For this task, I need ctime.

find has this baffling syntax of +x, x and -x for specifying 60 minutes, and it would read as "more than x", "exactly x" and "less than x", respectively. If we use the sequence `-ctime -60`, we'll get exactly the opposite of what we want; we'll get files that have been changed in the last 60 minutes.

This is a succinct way of solving this problem, and it's clear that a shell function to convert, say, bytes into megabytes easily can be produced as a one-liner.

Or is that what we are specifying? Without a unit indicated, the default time unit is really days, so -60 is actually files that have been changed in the last 60 days—not what we want!

To specify minutes, we want to use cmin rather than ctime (I told you find was confusing). Here's how that might look:

```
find . -cmin +60
```

The above also matches directories, however; so another predicate we'll want to add is one that constrains the results only to files:

```
-type f
```

(`type d` is only directories, and so forth).

But, that's not exactly right either, because we probably want to ensure that we only ever go one level deeper instead of spending a lot of time traversing a complex file tree. This is done with the little-used maxdepth parameter, which is described as "True if the depth of the current file into the tree is less than or equal to n." Now, let's put this all together:

```
find . -cmin +60 -type f -maxdepth 1
```

See how that all fits together?

Now, the last part of this requirement is actually to delete the matching file or files, and I have to admit that this gives me some cause for anxiety, because if you make even the slightest mistake with the find command, you can end up deleting tons of files you didn't want removed—not good. So, rather than just use `-delete`, I suggest you use `-print`, and for a day or so, let it run and have cron automatically e-mail the resulting report to you.

Hooking into Cron

Speaking of which, the way that you get to the data file that defines which jobs you want run when from the crontab facility is the crontab command. Log in as the desired user (probably root in this case), then type:

```
crontab -e
```

You'll now be editing a file with comments (lines starting with #) and lines composed of five space-separated values followed by an sh command, like this:

```
* * * * *  /home/taylor/every-minute.sh
```

This is rather brutal on the system. It invokes this script every single minute of every day—probably overkill for just about any process, but it illustrates the basic format of crontab entries.

The fields are, in order, minute, hour, day of month, month and day of year. To have our job run every hour, for example, we can simply set the minute field to a specific value. For example:

```
10  * * * *   /home/taylor/every-hour.sh
```

Every hour, at ten minutes after the hour, the script is run. That works.

Now, to stitch it all together, the best bet is to drop the find command into a very short shell script and invoke the script with cron, rather than having the command itself in the crontab file. Why? Because it gives you lots of flexibility and makes it very easy to expand or modify the script at any time.

Put everything in this column together and you should be able to really start exploiting some of the recurring job capabilities of your Linux box. I am a big fan of cron and have many, many jobs running on a nightly basis on my servers. It's well worth learning more about, as is the find command.

Now, what were we talking about earlier?■

Dave Taylor is a 26-year veteran of UNIX, creator of The Elm Mail System, and most recently author of both the best-selling *Wicked Cool Shell Scripts* and *Teach Yourself Unix in 24 Hours*, among his 16 technical books. His main Web site is at www.intuitive.com.

Introduction to SELinux, Part II

MICK BAUER

Understanding SELinux's security models is the first step in harnessing its power.

In my last column, we began exploring the concepts, terms and theory behind Security-Enhanced Linux (SELinux). This month, we conclude our overview, ending with a description of the SELinux implementation in Red Hat Enterprise Linux, Fedora and CentOS.

As much as I'd like to dive right in with the new material, SELinux is one of the most complex topics I've tackled in this column, so some review is in order. Rather than simply summarizing last month's column, however, here's a list of SELinux terms:

- Discretionary Access Controls (DACs): the underlying security model in Linux, in which every file and directory has three sets of access controls, known as permissions: one set each for user-owner, group-owner and other. These permissions can be changed arbitrarily, at the discretion of the file's or directory's owner
- Mandatory Access Controls (MACs): a much stronger security model, of which SELinux is an implementation, in which access controls are preconfigured in a system security policy that generally does not allow system users or processes to set or change access controls (permissions) on the objects they own.
- Subject: a process that initiates some action against some system resource.
- Action: a system function (writing a file, executing a process, reading data from a socket and so on).
- Object: any system resource (process, file, socket and so on) against which subjects may attempt actions.
- User: in SELinux, an SELinux-specific user account is separate from underlying Linux user accounts and owns or initiates a subject process.
- Role: analogous to Linux groups in that it represents a set of access controls that apply to a specific list of possible users. In SELinux, a user may be associated with multiple roles, but may assume (act within) only one role at a time.
- Domain: a combination of subjects and objects permitted to interact with each other.
- Type: synonymous with domain in SELinux.
- Security context: the user, role and domain/type associated with a given subject or object.
- Transition: when a process attempts to change from one role to another by spawning a new process that "runs as" the new role, or when a process attempts to create a new file or directory that belongs to a different role than its parent directory.
- Type Enforcement: the security model in SELinux in which processes are confined to domains via security contexts.

As I mentioned last time, Type Enforcement is the most important of the three security models implemented in SELinux. In fact, in the Red Hat Enterprise Linux (RHEL) targeted policy, which I cover at length later in this article, Type Enforcement is the *only* SELinux security model used.

Role-Based Access Controls

As important as Type Enforcement is, it's a very process-oriented model. It's most useful for "sandboxing" or isolating dæmons. But, what about actual human users, who may perform a variety of tasks on the system and, therefore, may need to traverse multiple domains?

SELinux's Role-Based Access Control (RBAC) model concerns the ways in which users may transition between the roles they're authorized to assume and, by extension, between the domains in which those roles have rights. In practical terms, such a transition occurs when a process running from within one domain spawns a process into a different domain.

For example, suppose user Mick is authorized to operate in the role Parent, which in turn is associated with the domains Supper and Bedtime. In order for Mick to transition from Supper to Bedtime (for example, to start a shell session in the Bedtime domain, with access to files and processes authorized for that domain but not for the Supper domain), an RBAC rule must explicitly allow the role Parent to transition from Supper to Bedtime. This is *in addition to*, not instead

of, the need for Parent to be defined in security contexts for those two domains.

Multi-Level Security

The third security model in SELinux is Multi-Level Security (MLS). MLS is in turn based on the Bell-LaPadula model for data labeling. The guiding principle of both the Bell-LaPadula model and MLS is "no read up, no write down". That is to say, a process (user) authorized to read data of one classification may not read data of a higher (more sensitive) classification, nor may that process (user) write data of a given classification anyplace in which it might be accessed by processes (users) authorized only to view data of lower (less sensitive) classifications.

For this model to work, each subject on the system must be associated with a security clearance—that is to say, the maximum sensitivity of data to which that subject may have access. Every file (object) also must be labeled with a classification that specifies the minimum clearance a subject must have in order to access it. The MLS Range field, supported in SELinux since Linux kernel 2.6.12, provides this information in the security contexts of both subjects and objects.

But, what about actual human users, who may perform a variety of tasks on the system and, therefore, may need to traverse multiple domains?

The traditional four data security classifications are, in decreasing order of sensitivity, Top Secret, Secret, Confidential and Unclassified. However, in MLS, many more such hierarchical classifications can be defined in your security policy. Also, each hierarchical classification can be associated with non-hierarchical compartments, which you can use to enforce a need-to-know policy in which subjects authorized at a given classification level may be granted access only to objects associated with specific compartments within that classification.

For example, suppose the process hamburgerd has overall subject clearance of Secret, and specific clearance (within the Secret classification) to the compartments ingredients and handshakes; such a clearance might be notated as { Secret / ingredients, handshakes }. If the file high_sign has an object clearance of { Secret / handshakes }, hamburgerd will be permitted to read it.

Note that by "non-hierarchical", I mean that compartments within the same classification are peers to each other. If I define two compartments, apples and oranges under the classification Classified, neither compartment is considered more sensitive than the other. However, any compartment associated with the Secret or Top Secret classification will be considered more sensitive than either { Confidential / apples } or { Confidential / oranges }.

Multi-Category Security

I've referred to SELinux's three security models (TE, RBAC and MLS). In Red Hat Enterprise Linux 5, there's actually a fourth: Multi-Category Security (MCS).

As you might imagine, the combination of hierarchical classifications and non-hierarchical compartments makes MLS well suited to large bureaucracies, such as military organizations and intelligence agencies, but too complex for more general purposes. Red Hat has therefore implemented an alternative file-classification model in RHEL 5's implementation of SELinux: Multi-Category Security (MCS).

MCS uses SELinux's MLS Range field, essentially by ignoring the Classification field (assigning a classification of 0 to all subjects and objects) and instead acknowledging only the Compartment field. In this way, the power of data labeling is simplified to something more like the Linux DAC group functionality. In other words, MCS is similar to MLS, but lacks the added complexity of hierarchical classifications.

SELinux Simplified: Red Hat's Targeted Policy

Now that you understand SELinux's underlying security models and are familiar with at least a portion of SELinux's formidable body of jargon, we can turn our attention to SELinux's debut in the mainstream: Red Hat's targeted policy.

For many if not most system administrators, having to understand SELinux's various security models and complex terms, and managing its myriad configuration files, which may cumulatively contain hundreds or even thousands of lines of text, makes tackling SELinux a highly unattractive undertaking. To address this problem, Red Hat devised a simplified SELinux

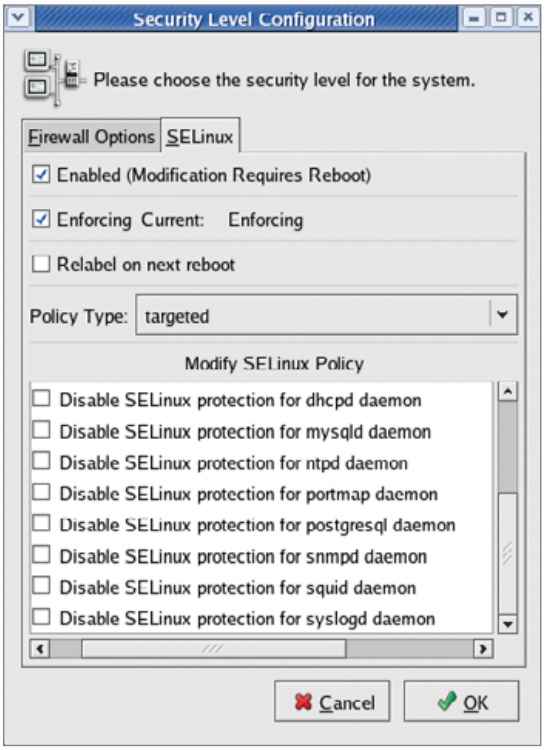


Figure 1. RHEL 4's system-config-securitylevel Tool

policy, called targeted, that emphasizes Type Enforcement, greatly simplifies RBAC and omits MLS altogether.

In fact, RHEL's targeted policy doesn't even implement Type Enforcement globally; it only defines domains for 12 specific subject dæmons, placing all other subjects and objects into a default domain, unconfined_t, that has no SELinux restrictions (outside of those 12 applications' respective domains).

The dæmons with SELinux domains in RHEL 4 and 5's targeted policy are:

- dhcpd
- httpd
- mysqld
- named
- nscd
- ntpd
- portmap
- postgres
- snmpd
- squid
- syslogd
- winbind

You may wonder, doesn't this amount to a global policy of "that which isn't expressly denied is permitted?" And, isn't that precisely backward of the "default-deny" stance that Mandatory Access Controls are supposed to provide?

Not really. It's true that the targeted policy falls well short of a trusted SELinux implementation of the kind you'd use for US Department of Defense work. However, neither does it amount to an "allow by default" policy. the regular Linux DAC (filesystem) controls still apply. So, if you think of the targeted policy as an extra set of controls layered *on top of*, not in lieu of, the normal filesystem permissions, application-level controls, firewall rules and other things you'd have on a hardened Linux system, you can see that even a limited SELinux policy can still play a meaningful role (no pun intended).

In fact, I'll go a step further and say that Red Hat's targeted policy is SELinux's best hope (to date) for mainstream adoption. Red Hat is by far the most popular Linux distribution to ship with any SELinux policy enabled by default; if that policy were locked down so tightly that any customized or substantially reconfigured application was barred from proper operation, most users would simply disable SELinux. (This was, in fact, what happened when Fedora Core 2 shipped with a "default-deny" SELinux policy.)

By enabling an SELinux policy that applies only to a limited, well-tested set of applications, Red Hat is minimizing

Resources

Faye and Russell Coker's article "Taking advantage of SELinux in Red Hat Enterprise Linux": **www.redhat.com/magazine/006apr05/features/selinux**

McCarty, Bill. *SELinux: NSA's Open Source Security Enhanced Linux.* Sebastopol, CA: O'Reilly Media, 2005. Definitive resource, but predates Red Hat and Fedora's implementation of targeted and strict policies.

Mayer, Frank, Karl MacMillan and David Caplan. *SELinux by Example: Using Security Enhanced Linux.* Upper Saddle River, NJ: Prentice Hall, 2007. Brand-new book, by several SELinux contributors.

Chad Hanson's paper "SELinux and MLS: Putting the Pieces Together": **selinux-symposium.org/2006/papers/03-SELinux-and-MLS.pdf**

"Red Hat Enterprise Linux 4: Red Hat SELinux Guide": **www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/selinux-guide/index.html**

Russell Coker's tutorial "Introduction to SELinux on Red Hat Enterprise Linux 4": **www.coker.com.au/selinux/talks/rh-2005/rhel4-tut.html**

the chances that a significant percentage of its users will associate SELinux with inconvenience and lost productivity. Furthermore, the targeted policy can be administered by a simple GUI, system-config-securitylevel, that doesn't require the user to know anything about SELinux at all.

The targeted policy ships with RHEL 4 and 5, Fedora Core 3 and later, and CentOS 4 and 5.

Red Hat's Strict Policy

The comprehensive "deny-by-default" policy originally developed for Fedora Core 2, called strict, is still maintained for RHEL, Fedora and CentOS, and it can be installed instead of targeted. However, strict is not officially (commercially) supported in RHEL due to its complexity. On most systems, this policy takes a lot of manual tweaking, both by editing the files in /etc/selinux and by using the standard SELinux commands chcon, checkpolicy, getenforce, newrole, run_init, setenforce and setfiles.

Note that Tresys (**www.tresys.com**) maintains a suite of free, mainly GUI-based, SELinux tools that are a bit easier to use, including SePCuT, SeUser, Apol and SeAudit. These are provided by RHEL's setools RPM package. Note also that on non-Red-Hat-derived Linux distributions, SELinux policies usually reside in /etc/security/selinux.

To customize and use the strict policy on RHEL 4, see Russell Coker's tutorial "Introduction to SELinux on Red Hat Enterprise Linux 4" (see Resources). You need to install the package selinux-policy-strict, available in Fedora's rawhide repository (the selinux-policy-strict package in Fedora Core 5 or 6 may also work in RHEL 4).

Conclusion

It's also possible, of course, to develop and enable your own SELinux policies from scratch, though doing that is well beyond the scope of this article. In fact, entire books have been written on this topic. See Resources for information on SELinux policy creation and customization.

And with that, I hope you're off to a good start with SELinux. Be safe!■

Mick Bauer (darth.elmo@wiremonkeys.org) is Network Security Architect for one of the US's largest banks. He is the author of the O'Reilly book *Linux Server Security*, 2nd edition (formerly called *Building Secure Servers With Linux*), an occasional presenter at information security conferences and composer of the "Network Engineering Polka".

Waysmall

An appliance approach is ideal for Asterisk.

JON "MADDOG" HALL

Another OpenBeach (**www.openbeach.org.br**) is over, and I sit in the Pousada Dos Golfinos for a few days both relaxing and writing business plans for various companies as well as this column.

This OpenBeach period was unusual due to the number of talks I gave to various organizations in Brazil, and also because I met a very interesting young man named Kristian Kielhofner and learned about his AstLinux project.

Unless you have been hiding under a rock for the last several years, you probably know about the FOSS project called Asterisk (**www.asterisk.org**), a complete PBX system for serving VoIP and wired telephone systems.

Asterisk works on many distributions of Linux, BSD and Mac OS X systems, and a lot of people simply put Asterisk on top of one of those systems, hook it up to the Internet and use it that way.

Some of the features Asterisk provides are caller ID, voice mail, direct inward dial (instead of having to ask an "operator" for an extension), call logging and accounting, support for a wide variety of codecs (the software that converts, compresses and massages voice signals into binary streams), conferencing, interactive voice response, call forwarding, interactive directory listing, music on hold and transfer, roaming extensions, remote call pickup, spell/say, and much, much more. Asterisk was started by a young man named Mark Spencer, who now is the president of a company called Digium (**www.digium.com**) that contributes to and supports Asterisk and makes systems that run it.

Kristian, on the other hand, decided that he would like to see how small a distribution of Asterisk and Linux he could make and still have full functionality. He calls it AstLinux (**www.astlinux.org**).

Small, tailored distributions have more advantages than merely saving disk space. By eliminating functionality not used and using libraries and programs optimized for size, you often can increase the security and (by better utilization of both main memory and cache) the speed of systems with relatively low-speed CPUs. You also can reduce the power requirements required by systems that, by their nature, should remain on (as a phone system should), and by reducing moving parts (such as disks and fans), you increase the system's life expectancy. Here at the beach (or on board boats), the salt air is very destructive of electronics, but those with moving parts, such as disks and fans, are particularly susceptible.

Although AstLinux is a compact distribution of Asterisk, Kristian points out that it is full-featured, able to be tailored, and even has some other VoIP applications, such as OpenSER.

Because AstLinux is aimed at small, embedded systems, it has specific images for processor architectures other than the traditional Intel, and specifically for low-power processors. It is also designed to run from Compact Flash or boot live from a small business-card-sized CD.

In addition to introducing me to AstLinux, Kristian introduced me to Gumstix.

Now, I have always been interested in the very large and the very small. I like Linux high-performance clusters (or huge data farms) and embedded systems. I have purchased (yes, I normally buy them with my own money) more embedded system development kits than grains of sand on the beach, but I always like seeing what else is out there. Somehow I had missed Gumstix!

Gumstix computers (**www.gumstix.com**) are based on the Intel XScale processor and come in 200MHz or 400MHz models. They are literally about the size of a stick of a famous-name chewing gum and are expandable by adding on similarly sized option boards. The basic Gumstix can have Bluetooth capability right on the board, and with other option boards, they can be turned into either a Waysmall computer (with serial ports and a USB client) or a Netstix computer (with Ethernet and a Flash socket), including a nice case for each.

With 64MB of RAM and 16MB of Flash, it is obvious that these systems are not general-purpose gaming desktop replacements. On the other hand, my first UNIX

"workstation" had 1MB of RAM and a 10MB disk (and yes, we did have the X Window System on it), so with careful pruning of code, you can put a reasonable amount of functionality on the Gumstix systems. And, with either wired Internet or the new 802.11g wireless option, you could create a diskless client for even more capabilities.

Another nice thing about Gumstix is that the technical information for its boards are available under a Creative Commons Attribution-ShareAlike 2.5 License. For those who want to design their own circuits to use with Gumstix, this is invaluable.

One caution, however. Although the Gumstix site is fairly helpful with all sorts of FAQs and other information, it does not provide step-by-step instructions.

Of course, there are other embedded systems and system suppliers. The PC/104 (**www.pc104.org**), the uClinux Project (**www.uclinux.org**) and SSV in Germany (**www.ssv-embedded.de/ssv/english/products/htm**) also are interesting in their own right. It is just that the Gumstix line is so darn cute!

Finally, I would like to say a few words about Kristian himself. I spent quite a few days with Kristian as we gave talks in Saõ Paulo, Brasilía, Curitiba and, finally, Florianópolis, Brazil. Some of Kristian's talks were technical, but as quite a few were given with high-school or college students in the audience, Kristian took some time to discuss how FOSS had affected his life.

Kristian gave credit to FOSS and the FOSS community for giving him the opportunity to turn his life around and have fun learning and building both a project and a business based on FOSS. I have met many young people like Kristian, but it is always nice to meet another, and particularly one that is as friendly and outgoing as Kristian. It validates what the FOSS community has been saying all along—that FOSS can help build character and open opportunities for people of all ages.

As I finish my last Caipirinha, and the sun goes down, I think of projects that could use a small system....■

Small, tailored distributions have more advantages than merely saving disk space.

Jon "maddog" Hall is the Executive Director of Linux International (www.li.org), a nonprofit association of end users who wish to support and promote the Linux operating system. During his career in commercial computing, which started in 1969, Mr Hall has been a programmer, systems designer, systems administrator, product manager, technical marketing manager and educator. He has worked for such companies as Western Electric Corporation, Aetna Life and Casualty, Bell Laboratories, Digital Equipment Corporation, VA Linux Systems and SGI. He is now an independent consultant in Free and Open Source Software (FOSS) Business and Technical issues.

DOC SEARLS

DIY Internet Infrastructure

An open-source angle on muni-Net infrastructure build-out.

We want something that works for everybody: a new infrastructural tide that lifts all boats—like Linux did, and still does.

Although Moore's Law drives up compute power, network speeds languish. One reason I moved to Santa Barbara in 2001 was that Internet connectivity was much better here than in Silicon Valley, where I had lived for the preceding 16 years. Six years later, however, my connection speeds in Santa Barbara are barely any better, and the costs have gone up. Worse, the city appears to be a low-priority region for Verizon and Cox, our local telco/cableco duopoly. Although Verizon is rolling out fiber to homes in dozens of communities around the country, Santa Barbara isn't one of them. And, although Cox is also rolling out higher-speed services, Santa Barbara is smaller and farther away from Cox's Atlanta headquarters than any other markets.

And, we're not alone. The same kind of story is happening in most communities across the US. This is why hundreds of those communities are doing what Linux and open-source developers have done all along: taking matters into their own hands, doing for themselves as individuals and groups what proprietary businesses and industries can't or won't do. They're rolling their own infrastructure.

That's why I've been taking part in a local citizen effort to build infrastructure that the local duopoly can't or won't provide. This has involved lots of meetings and discussions—not only amongst ourselves, but with local businesses, civic leaders, elected representatives, infrastructure (fiber, wireless) deployment contractors and other folks, including cable and phone company people. We want something that works for everybody; a new infrastructural tide that lifts all boats—like Linux did, and still does.

This is why our emphasis is more on fiber than on wireless. We want Net infrastructure that maximizes capacity for the longest possible time, and for the largest number of people and locations—much as we might be doing if the challenge were building roads or reservoirs.

It's still early, and we're pioneering here, as are many other communities. To help gather useful thinking at this pioneering stage, I've put together a FAQ for myself, and for anybody else who wants the benefit of the same thinking. Because much of this thinking is informed by, and modeled on, the successful experiences of the Linux and Open Source movements, I thought I'd share my FAQ draft here.

Bear in mind that this is just thinking out loud at this point. It's provisional. I offer it in hope that it's useful to other folks in other towns who are trying to do the same kind of thing. I also offer it in faith that the smart readers of *Linux Journal* will help myself and others think and work our way to local Net infrastructure that favors everybody, and not just incumbent carriers.

So here goes.

Q Why does <your town or region here> need fiber-optic Internet infrastructure?

A For the same reason it needs roads, water, waste treatment and electricity. The Internet is becoming a "fifth utility", no less essential than those other four. Fiber-optic cabling is the best form of "pipe" for carrying the Net. It is also the most "green" and has enormous capacity. One fiber optic pair can carry thousands—even millions—of times more data than any form of copper wiring or any form of wireless connection.

Q Why not do wireless infrastructure instead of fiber?

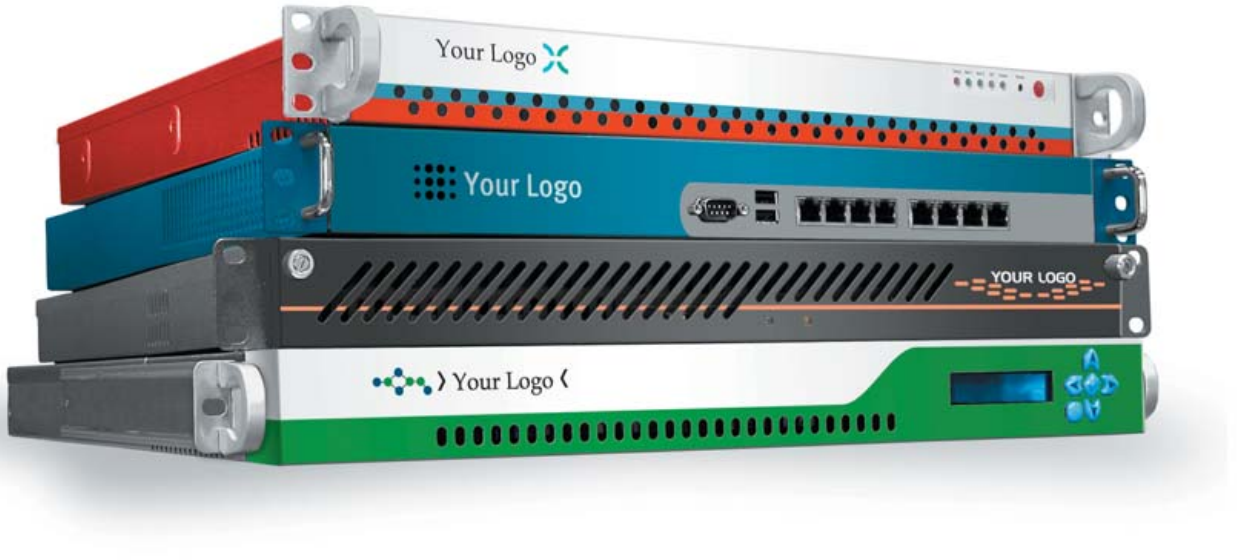
A Even wireless needs a fiber backbone. We think it's essential to bring the best connections to everybody. We also can add lots of wireless service on top of fiber infrastructure. Even cities opting for wireless build-outs require fiber backbones. We want to make the Net's maximum capacity available to everybody. This requires bringing fiber connections to homes, businesses, hospitals, libraries, government offices and other fixed locations—or as close as we practically can. After that we can easily add all the wireless or wired coverage we want.

Q Isn't the Internet just a service we already get from cable or phone companies?

A No. The Internet is not itself a "service". It's a means for transporting data between devices anywhere—without regard for the distance between them or the routes between them. It also was designed on "end-to-end" principles, which make the "middle" of the network as transparent, uncomplicated and cost-free as possible.

The Internet is radically different from telephone, cable TV and other networks optimized for a single purpose. It exists to support whatever anybody wants to build on it or use it for. Its purpose is to maximize support for a maximum variety of uses, while minimizing its own intrinsic costs. In this respect, it is a public utility like water, roads, waste treatment and electric power. The Internet cares no more about how you use it than your electric outlets care what you plug in to them or than your water faucet cares about whether you're washing dishes or filling a cup.

The Industry Leader for Server Appliances



Custom server appliances or off the shelf reference platforms, built with your image and software starting under $1,000.

From design to deployment, we handle it all.

Delivering an appliance requires the right partner. MBX Systems is the right partner. We understand that by putting your name on our hardware, you're putting your reputation in our hands. We take that seriously. We provide the services you need to support your customers. Better than the competition. You never even need to touch the hardware. Engineering. Design. Deployment. We handle it all, so you can focus on what's important to you. Your software. Your sales. Your success.

Visit us at www.mbx.com or call 1-800-681-0016 today.

MBX systems

www.mbx.com | 1-800-681-0016

We want to make the Net's maximum capacity available to everybody.

Q But aren't telephone and cable companies utilities as well?

A Yes, but they are also businesses, and they're designed originally for single purposes. Although the Net today is carried to our homes and businesses by these two businesses, the relationship in the long run is the other way around. Telephony and video used to be analog services. Now they are only two among many other forms of data that the Net is capable of transporting. So in the long run, we need to see telephony and video as two among a countless variety of businesses that are supported by the Net. And, we need to stop seeing the Net as gravy on top of telephone and cable service.

Q Is Net infrastructure costly to deploy?

A The Internet itself is not inherently scarce—as are water, electricity and roads. Instead, the Internet was designed as a way to remove distance between every connected device and the people who use those devices. There is no "long distance" to the Internet. Once a device is "on" the Net, it is zero distance from every other device in the world that's also on the Net. That's because the Net was designed so the functional distance between any two keyboards in the world is no greater than the distance between those keyboards and their screens.

Meanwhile, the cost of digging trenches, pulling cable and other expenses is far above zero. But it is also far less than the cost of building roads, water distribution and waste treatment infrastructures. And in the long run, it is just as important as any of those.

Q Doesn't the Net cost money to maintain?

A Yes, it does. But the maintenance costs are more like those for roads than for phone or cable TV. Unless a town (or a county) wants to go into the phone and TV businesses, the engineering and maintenance costs do not have to be high. But capable engineers and maintenance workers are required, just as they are for other utilities.

Q Isn't the Internet service we get from telephone and cable companies good enough? We have e-mail. We can browse the Web. We can do instant messaging. We can search on Google and buy stuff from Amazon.

A The phone and cable companies deserve credit for offering Internet services and contributing to the expanded reach of the Net. But it is important to understand two things. First, the Net is far more than just an extra service provided by telephone and cable companies—even though that's how those companies sell and bill for it. Second, the Internet is a third-priority offering for both telephone and cable companies. Right now telcos and cablecos are working much harder at getting into each other's core businesses than they are at expanding Internet services. And, the Net is about much more than e-mail, searching and browsing. It is becoming an essential backbone for civilization itself. There are many public and private service needs that can be served only by reliable high-capacity Net infrastructure. Doctors can do diagnostics and even surgery over distances. Public safety services can communicate and share data rapidly. Businesses that deal in graphical imagery, high-quality audio and moving pictures can produce and share their work far more easily. Most important, businesses and civic activities of all kinds can be supported—not only the two incumbent businesses that first brought the Net to our homes.

Q Doesn't all this threaten telephone and cable companies?

A It doesn't have to, because there is nothing keeping telephone and cable companies from also going into other businesses that are opened up by a growing Internet. In other words, there are benefits to incumbency other than leveraging original business models.

In fact, the Net offers enormous opportunities to telephone and cable companies—opportunities to provide countless services in addition to their traditional ones. That's because there is no limit to what you can do with the Net. Telephone and cable companies have a head start, both in existing facilities and existing relationships with thousands or millions of customers. Instead of being in one, two or three businesses (the phone, cable and Net "triple-play" businesses they're in now), these companies can offer an infinite variety of value-added services to individuals and companies and organizations that use the Net.

Q Why do cities and counties need to build out Net infrastructure? Why not let the marketplace take care of it?

A Unfortunately, the marketplace we have is not a free and open one. Telephone and cable companies are captive to long-standing regulatory environments that are changing very slowly. Also, neither side has shown much interest in putting Net build-out ahead of their core businesses. Both telephone and cable operators have powerful lobbying forces in Washington, DC, and at the state level as well, working to protect their traditional businesses from the "threat" of Internet growth. Giving them exclusive rights to control Internet infrastructure and growth is a guarantee that the going will be very slow.

Q Doesn't the federal government care?

A Not much. Although the Internet was born in the US, neither our federal government nor its protected communications duopoly have shown much interest in the Net's development. Nor has our government paid much attention to how well the Internet supports and sustains economic growth. Other countries—Korea, Japan, Denmark, Netherlands and France, for example—have encouraged Net build-out for most of the last decade. As a result, the US is now 11th in "broadband" penetration. By deciding to leave Net

StorageWare SA108
-Supports the latest Intel® Quad-Core Xeon® Processors
-Intel 5000P Chipset with 1333Mhz FSB
-Up to 16GB of PC2700 ECC Registered DDR Memory
-Up to 2.25TB Redundant Storage Capacity (3TB Raw)

Starting at $2899

PerformanceWare 1550
-Supports the latest Intel® Quad-Core Xeon® Processors
-Intel 5000P Chipset with 1333Mhz FSB
-Up to 16GB of PC2700 ECC Registered DDR Memory
-Up to 584GB High Performance SAS Redundant Storage

Starting at $3999

StorageWare SA360i
-Supports the latest Intel® Quad-Core Xeon® Processors
-Intel 5000P Chipset with 1333Mhz FSB
-Up to 16GB of 667MHz ECC Registered DDR2 Memory
-Up to 11.25TB of Native Redundant Storage

Starting at $4999

We've expanded our StorageWare server line featuring the latest Intel® processors including Core™ 2 Duo, Dual-Core or Quad-Core Xeon® and Serial ATA RAID 6 to bring you the fastest and most reliable disk array available. With over 11TB of storage in a RAID 6 configuration, these systems showcase high-performance, high-density enterprise storage that keeps your data safe.

pogo [linux]™

Experience, Imagination, and Support.
Pogo Linux is Hardware Solutions Built for Linux, Built for You.

www.pogolinux.com

intel Xeon® inside™
Quad-core. Unmatched.

To get started, contact us at 888.828.POGO or inquiries05@pogolinux.com
Pogo Linux, Inc. 701 Fifth Ave. Suite 6850, Seattle, WA 98104

And, the Net is about much more than e-mail, searching and browsing. It is becoming an essential backbone for civilization itself.

build-out to the monopolists who feel threatened by it, our federal and state governments have assured limits on our economic growth and have reduced our competitiveness in the world marketplace.

Q How have the carriers crippled our Internet service?

A The Internet was designed originally as a symmetrical system. That means the "upstream" and "downstream" speeds should be the same. That's the kind of Internet connectivity we find in universities and inside large companies. But it's not what the telephone and cable companies provide to our homes and small businesses. What we get is asymmetrical—much higher downstream than upstream. The reasons are not necessarily bad ones. Most of us consume far more data than we produce. This is especially true when we download large graphical files, watch a YouTube video or listen to the live stream of a radio station over the Net. The carriers have optimized their systems for asymmetries between production and consumption.

The problem is, these asymmetrical lines relegate everybody to a consumer role and prevent us from becoming producers as well. This limitation is compounded by what are called "port blockages". This is where our phone or cable company prevents us from setting up our own Web server or running our own mail server. Again, they have some good reasons for blocking the ports on our computers that those services could run on. Spammers, for example, can take advantage of open mail server ports on our computers. But these port blockages also prevent all types of uses, including the ability to set up home businesses of many kinds.

So, instead of, say, offering services that aid in the development of small and home businesses, the carriers just shut off possibilities to avoid hassles that might distract focus from their core phone and cable TV businesses.

There is also, in both the telephone and cable businesses, a traditional high-charge orientation toward business customers. If you're a business, they want to charge you a lot more money for the same level of service provided to "consumers" who produce nothing other than a monthly payment. The carriers will talk about providing a higher grade of service for the money, but the costs are often so high that they drive businesses away. As a result, businesses don't take advantage of what the Net has to offer. They buy the lowest-price offering and stick to browsing the Web and doing e-mail.

Q Isn't local infrastructure build-out a case of government competing with private industry?

A No. It's a case of citizens finding a way to do what a protected duopoly cannot. What we are doing is also not competitive. We want to open our new fiber infrastructure to use by anybody, including cable and phone companies. We have their interests at heart too. By building out pure Net infrastructure—rather than competing with cable TV and phone systems—we are protecting and supporting their core businesses.

Q Isn't this different from what we're seeing in other cities that roll out "triple-play" fiber systems?

A Yes, it is. By just installing the "transport", and leaving the services up to other parties (including the phone and cable companies), we are doing two things. First, we're saving money by not getting into businesses and facilities that actually cost a lot to start and maintain. Second, we are treating the Net as the simple utility it needs to be. As a city, we know how to maintain roads, water treatment and waste facilities, and so on. We also know how to support businesses rather than compete with them. That's what we are doing here.

Q What will it cost to bring fiber-optic cabling to homes and businesses?

A We have estimates of up to $2,500 per "drop"—about the cost of a big flat-screen TV (without the immediate and certain depreciation). But the cost of fiber-optic cabling itself is actually less in most cases than the labor cost of burying it in the ground or hanging it from poles. There is much we can do to drive costs down for the city and to make it easy for citizens to connect to the network as they need it. For example, we can require that conduit and fiber be put in the ground everywhere a trench is dug. We can hang fiber on poles first and work out the individual drops later. We can create a regulatory and procedural environment that encourages citizens and local businesses to build their own neighborhood fiber networks, which we can then connect to our publicly or privately own owned "backbone" whenever they're ready. We would love to create an environment where local businesses do much of the build out and make money in the process.

Q If you're not selling phone and cable TV services, how are you going to pay back the cost of installing the fiber cabling?

A We'll have to approach this the same way we approach building out any new public infrastructure—just as we did when we built our water systems, our waste treatment plants, our roads. We need to think creatively about this, but also take advantage of what we already know about building utility infrastructure.

Q What do you expect will be the economic benefits of this?

A We can begin to hint at those by looking back at what the Internet has done for our lives, and our economies, over the last ten years or more—and multiplying it by countless citizens and businesses that are finally able to do what they want with a Net infrastructure that truly supports it.

Q Are there any downsides to building this out?

A No. And that's the real bottom line.■

Doc Searls is Senior Editor of *Linux Journal*. He is also a Visiting Scholar at the University of California at Santa Barbara and a Fellow with the Berkman Center for Internet and Society at Harvard University.

SUPERMICRO®

1U Twin™ Two DP Systems in 1U

up to 16 cores Xeon in 1U

SuperServer 6015T-T/INF

- **Twin** Intel® Quad-Core/Dual-Core CPU
  Max. up to 4 Xeon® 5300/5100 processors
- **Twin** Dual CPU platforms with Intel 5000P/1333/1066MHz
- **Twin** 8 DIMM supports 64GB of Fully-Buffered DIMM 667/533
- **Twin** PCI-E x8 (low-profile) I/O slots
- **Twin** 2 x 3.5" hot-swap SATA drive bays
- **Twin** Dual LAN Gigabit LANs/Infiniband
- **Twin** VGA ports with 16MB PCI Graphic
- **Twin** rear I/O ports
- **Twin** 3 sets heavy-duty counter-rotating fans
- **Twin** Management socket for IPMI 2.0 with virtual media over LAN and optional KVM-over-LAN
- **980W** High-efficiency power supply w/I^2C (90%+*)

Advantages & Benefits

- **16 cores in single 1U** - Support heavy network traffic & computing intensive applications
- **Double density and computing power** - Fit up to 84 servers in a 42U standard rack & save IT space rental costs
- **Independent power control** - Each node has its own power front-panel
- **Independent cooling control** - Each node has 3 heavy duty fans with optimal fan speed control
- **Higher power utilization increases power supply efficiency** - Saves energy costs
- **Two nodes in one 1U** - Extended server life cycle, saves chassis, power supply and rack costs
- **Reduce power cables and power strips** - For easy cabling, better airflow and reduced cabling costs
- **Save maintenance / management costs** - Maintain two systems logically in one physical space

intel Xeon® inside™

Applications

- HPC cluster computer nodes, data center, data farm, front-end server and other computing intensive applications

* Our new generation power supply efficiency is 90% in a typical loading operation

AMAX	Arrow Electronics	ASI	Bell Micro	Ingram Micro	MA LABS	Synnex	Tech Data
1-800-800-6328	1-888-427-2250	1-800-2000-ASI	1-800-232-9920	1-800-456-8000	1-408-941-0808	1-800-756-5974	1-800-237-8931
www.amax.com	www.arrownacp.com	www.asipartner.com	www.bellmicro.com	www.ingrammicro.com	www.malabs.com	www.synnex.com	www.techdata.com

TECH TIPS

Make KPilot work with Ubuntu, watch processes, lock files and create passwords in PHP.

Making KPilot Work with Ubuntu

The version of KPilot that comes with Ubuntu/Kubuntu Edgy Eft does not work with many Palm devices, such as the Palm TX, many Treo phones and others. Although many people are complaining about the difficulty of setting up the USB port, this tip doesn't address that particular issue. Even if you get the USB port working (I use a network sync, so it doesn't matter to me), you'll encounter other problems. For example, in many cases, KPilot copies the records from your Palm but erases the records from your Palm in the process.

At the time of this writing, the good folks at Ubuntu have not yet seen fit to update KPilot. There's no need to wait though. You can download the latest KPilot, compile and install it yourself. The version I downloaded works fine with my Palm devices.

You must have the KDE and Qt development libraries to compile KPilot, so you will need to install kde-devel at the very least. You also need cmake, which isn't installed by default in Ubuntu. You need to install the latest version of pilot-link separately as well, and compile it, first. In this example, I installed pilot-link in /usr/local/src/pilot-link-0.12.1. I also set the following environment variables for my platform (this is optional and may not apply to your platform):

```
export CFLAGS="-march=athlon64 -O2 -pipe"
export CXXFLAGS="${CFLAGS}"
export CPPFLAGS="${CFLAGS}"
export CXX="g++"
```

Here are the commands to download, make and install KPilot:

```
cd /usr/local/src
svn co svn://anonsvn.kde.org/home/kde/branches/KDE/
➥3.5/kdepim/kpilot/
cd kpilot
./configure --prefix=/usr --with-pilot-link=/usr/
➥local/src/pilot-link-0.12.1
make -f Makefile.cmake
make -f Makefile.cmake install
```

If you already have KPilot running, you need to exit, and you may even have to kill the dæmon with the command:

```
killall kpilotDaemon
```

Restart KPilot, and now you should be able to sync without problems. If you want to keep up to date with the latest changes, you can update the source code with the following command (obviously, you need to configure and install again afterward):

```
svn update kpilot
```

—Nicholas Petreley

Linux watch Command

Much open-source software for Linux has good monitoring commands for observing process activity. Some of the commands do not have graphical user interfaces, and in other cases, administrators prefer to use the command line. Monitoring the progress of an activity is a continuous task.

The Linux watch(1) command (**linux.about.com/library/cmd/blcmdl1_watch.htm**) is a useful tool for monitoring progress. It allows users to run a command and watch the output in a terminal window. It can execute the monitoring command at regular intervals and show differences (option -d) between successive updates.

Many Amanda (**amanda.zmanda.com**) users, myself included, use the watch command to observe the Amanda backup progress. The Amanda status command amstatus (**wiki.zmanda.com/index.php/amstatuscommand**) can be run with the watch command every minute to monitor the progress for each filesystem being backed up:

```
watch --differences=cumulative
➥--interval=60 amstatus backupconfig
```

The above command watches the backup progress for the Amanda configuration backconfig.

Another use for watch is to watch memory usage in a system:

```
watch cat /proc/meminfo
```

—Paddy Sreenivasan

PHP Create Password Script

The following tip comes courtesy of *Foundations of PEAR: Rapid PHP Development* by Nathan A. Good and Allan Kent, published by Apress (**www.apress.com/book/bookDisplay.html?bID=10181**).

This tip shows how to generate strong passwords using a PEAR package called Text_Password. To use the code shown in this tip, you need to have PEAR installed along with PHP, and you need to install the Text_Password package. To install the Text_Password package, type:

```
pear install text_password
```

The Code:

The PHP script that creates the password looks like this:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Password Example</title>
```

```
<style>
    li {
        font-style: italic;
    }
</style>
</head>
<body>
<?php
// require the Text_Password package
// to be included in the page
require_once "Text/Password.php";
?>
<p>
<strong>Here is a pronounceable password, defaulting
        to 10 characters:</strong>
<br />
<em><?php echo Text_Password::create(); ?></em>
</p>
<p>
<strong>Here are 5 unpronounceable passwords, with
        a length of 15 characters each:</strong>
<br />
<ul>
<?php
    $passwords = Text_Password::createMultiple(5,
    15, 'unpronounceable');
    foreach ($passwords as $password) {
    ?>
    <li><?php echo $password; ?></li>
    <?php
    }
?>
</ul>
</p>
</body>
</html>
```

The Results:

When the script is executed, it generates output very similar to that shown here. Of course, because the passwords are generated randomly, your actual results will differ slightly:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Password Example</title>
<style>
    li {
        font-style: italic;
    }
</style>
</head>
<body>
<p>
<strong>Here is a pronounceable password, defaulting
  to 10 characters:</strong>
<br />
<em>vumaechoud</em>
</p>
<p>
<strong>Here are 5 unpronounceable passwords, with a
  length of 15 characters each:</strong>
<br />
<ul>
    <li>E_3uYlRxYY2n%pd</li>
        <li>Ghn0Q@XZr%DBvDe</li>
        <li>0tAUoGoJR7C1zo2</li>
        <li>f#EA5jHIZmjaW8O</li>
        <li>1cbc7fhL@d#RHWM</li>
    </ul>
</p>
</body>
</html>
```

How It Works:

The Text_Password::create() method can be called statically. It returns a pronounceable password with a default length of ten characters, as shown in the output.

The Text_Password::createMultiple() method, also called statically, can accept parameters that allow you to specify how many passwords you want returned, the number of characters in each password and that they be either pronounceable or unpronounceable, depending

on the desired complexity for the password. Passwords that are unpronounceable have numbers and punctuation marks in them.

Using the Text_Password package, you quickly can write PHP scripts that allow your application or Web site to have the capacity to generate passwords.

—Nathan A. Good and Allan Kent

Locking Files

This tip comes courtesy of *Linux Journal* columnist Dave Taylor and No Starch Press.

Any script that reads or appends to a shared data file, such as a log file, needs a reliable way to lock the file so that other instantiations of the script don't step on the updates. The idea is that the existence of a separate lockfile serves as a semaphore, an indicator that a different file is busy and cannot be used. The requesting script waits and tries again, hoping that the file will be freed up relatively promptly, denoted by having its lockfile removed.

Lockfiles are tricky to work with though, because many seemingly foolproof solutions fail to work properly. For example, the following is a typical approach to solving this problem:

```
while [ -f $lockfile ] ; do
  sleep 1
done
touch $lockfile
```

Seems like it should work, doesn't it? You loop until the lockfile doesn't exist, then create it to ensure that you own the lockfile and, therefore, can modify the base file safely. If another script with the same loop sees your lock, it will spin until the lockfile vanishes. However, this doesn't in fact work, because although it seems that scripts are run without being swapped out while other processes take their turn, that's not actually true. Imagine what would happen if, right after the done in the loop just shown, but before the touch, this script were swapped out and put back in the processor queue while another script was run instead. That other script would dutifully test for the lockfile, find it missing and create its own version. Then, the script in the queue would swap back in and do a touch, with the result that the two scripts both would think they had exclusive access, which is bad.

Fortunately, Stephen van den Berg and Philip Guenther, authors of the popular procmail e-mail filtering program, include a lockfile command that lets you safely and reliably work with lockfiles in shell scripts.

Many UNIX distributions, including Linux and Mac OS X, have lockfile already installed. You can check whether your system has lockfile simply by typing `man 1 lockfile`. If you get a man page, you're in luck! If not, download the procmail package from **www.procmail.org**, and install the lockfile command on your system. The script in this section assumes you have the lockfile command.

The Code:

```
#!/bin/sh

# filelock - A flexible file locking mechanism.

retries="10"             # default number of retries
action="lock"            # default action
nullcmd="/bin/true"      # null command for lockfile

while getopts "lur:" opt; do
  case $opt in
    l ) action="lock"      ;;
    u ) action="unlock"    ;;
    r ) retries="$OPTARG"  ;;
  esac
done
shift $(($OPTIND - 1))

if [ $# -eq 0 ] ; then
  cat << EOF >&2
Usage: $0 [-l|-u] [-r retries] lockfilename
Where -l requests a lock (the default), -u requests
an unlock, -r X specifies a maximum number of retries
before it fails (default = $retries).
EOF
  exit 1
fi

# Ascertain whether we have lockf or lockfile
# system apps

if [ -z "$(which lockfile | grep -v '^no ')" ] ; then
  echo "$0 failed: 'lockfile' utility not
  found in PATH." >&2
  exit 1
fi

if [ "$action" = "lock" ] ; then
  if ! lockfile -1 -r $retries "$1" 2> /dev/null;
    then echo "$0: Failed: Couldn't create
    lockfile in time" >&2
    exit 1
  fi
else    # action = unlock
  if [ ! -f "$1" ] ; then
    echo "$0: Warning: lockfile $1 doesn't
    exist to unlock" >&2
    exit 1
  fi
  rm -f "$1"
fi

exit 0
```

Running the Script:

Although the lockfile script isn't one that you'd ordinarily use by itself, you can try to test it by having two terminal windows open. To create a lock, simply specify the name of the file you want to try to lock as an argument of filelock. To remove the lock, add the -u option.

The Results:
First, create a locked file:

```
$ filelock /tmp/exclusive.lck
$ ls -l /tmp/exclusive.lck
-r--r--r--  1 taylor  wheel  1 Mar 21 15:35
 ➥/tmp/exclusive.lck
```

The second time you attempt to lock the file, filelock tries the default number of times (ten) and then fails, as follows:

```
$ filelock /tmp/exclusive.lck
filelock : Failed: Couldn't create lockfile in time
```

When the first process is done with the file, you can release the lock:

```
$ filelock -u /tmp/exclusive.lck
```

To see how the filelock script works with two terminals, run the unlock command in one window while the other window spins trying to establish its own exclusive lock.

Hacking the Script:
Because this script relies on the existence of a lockfile as proof that the lock is still enforced, it would be useful to have an additional parameter that is, say, the longest length of time for which a lock should be valid. If the lockfile routine times out, the last accessed time of the locked file could then be checked, and if the locked file is older than the value of this parameter, it safely can be deleted as a stray, perhaps with a warning message, perhaps not.

This is unlikely to affect you, but lockfile doesn't work with NFS-mounted disks. In fact, a reliable file-locking mechanism on an NFS-mounted disk is quite complex. A better strategy that sidesteps the problem entirely is to create lockfiles only on local disks.

Excerpted with permission from the book *Wicked Cool Shell Scripts: 101 Scripts for Linux, Mac OS X, and UNIX Systems* by Dave Taylor, published by No Starch Press (**www.nostarch.com/wcss.htm**).

Credits

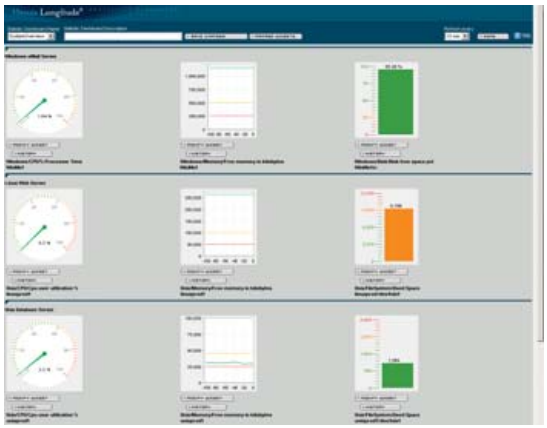
- Paddy Sreenivasan is VP of engineering and cofounder of Zmanda.
- Nathan A. Good lives in the Twin Cities area in Minnesota. He is a software developer, system administrator and author. He has written and co-written several books and articles on open-source technologies.
- Allan Kent is a born and bred South African and still lives and works in Cape Town. He has been programming in various languages and on diverse platforms for more than 20 years. He's currently the head of technology at Saatchi & Saatchi Cape Town.
- Dave Taylor is a longtime UNIX and Linux geek and runs the popular **www.AskDaveTaylor.com** tech support blog. His book *Wicked Cool Shell Scripts* can be found at **www.intuitive.com/wicked** and the entire library of scripts at **www.intuitive.com/wicked/wicked-cool-shell-script-library.shtml** .■

Linux Journal pays $100 for reader-contributed tech tips we publish. Send your tips and contact information to techtips@linuxjournal.com.

Heroix's Longitude

Heroix just announced version 4 of Longitude, the firm's agentless performance monitoring and reporting software. The application is intended to monitor an enterprise's entire IT infrastructure, including OS, Web, database, J2EE, messaging, infrastructure, and user and business metrics, all out of the box and without agent software on monitored computers. New features in version 4 are a new centralized event monitor; a customizable, real-time statistics dashboard; expanded monitoring coverage to any SNMP-based device via a graphical studio; an archived report portal; and consolidation of Windows Event logs. Platforms monitored are Red Hat, SUSE Linux, Solaris, HP-UX, AIX and Windows. A 14-day free trial is available at Heroix's Web site.

www.heroix.com



eZ Systems' eZ Publish

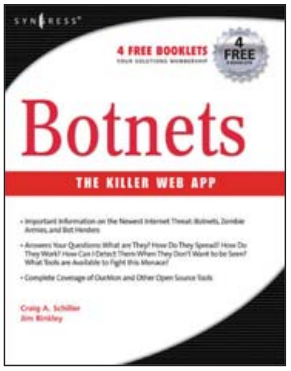
eZ Systems recently released version 3.9 of eZ Publish, the company's open-source, ready-to-run enterprise management system and framework. eZ Publish is an application for creating Web sites, on-line stores, intranets and extranets. New features in 3.9 include an intuitive front-end editor (Web site interface), enhanced shipping manager, extended LDAP user-group mapping, support for translating class attribute names and more. The product is available in either out-of-the-box or tailor-made solutions for the varying needs of clients. GPL'd Linux and Windows versions are available for download at the eZ Systems' Web site.

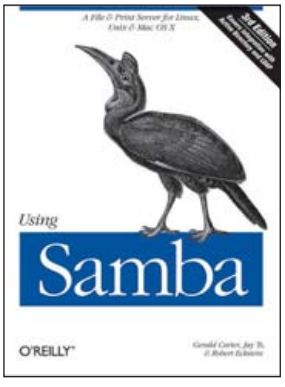
www.ez.no

Craig A. Schiller and Jim Binkley's *Botnets: The Killer Web App* (Syngress)

If battling botnets gives you nightmares, the new book *Botnets: The Killer Web App* may slay those pesky nocturnal dragons in your head. The book's mission is to arm you with useful information on botnets, zombie armies and bot herders, answering questions, such as "What are they? How do they spread? How do they work? How can I detect them when they don't want to be seen? What tools are available to fight this menace?" The main tools covered here are open source, with a focus on the network monitoring tool Ourmon.

www.syngress.com



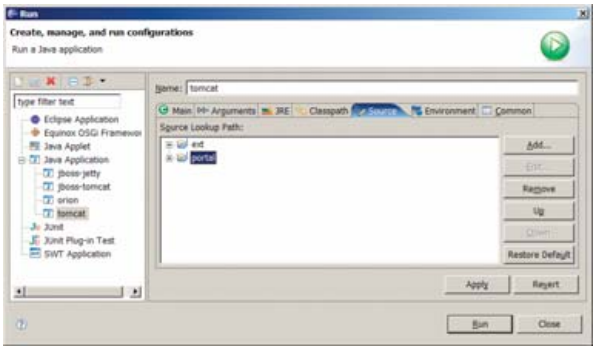


Gerald Carter, Jay Ts and Robert Eckstein's *Using Samba* (O'Reilly Media)

In our community, Samba is the way to go to turn your Linux or UNIX system into a file-and-print server for Windows network clients. Now in its third edition, *Using Samba* is the book "officially adopted by the Samba team" says the publisher. Further, it "delves into the internals of the Windows activities and protocols to an unprecedented degree, explaining the strengths and weaknesses of each feature in Windows domains and in Samba itself." This third edition covers Samba 3.x features, such as integration with Active Directory and OpenLDAP, migrating to Samba from Windows NT 4.0 domains, delegating administrative tasks to non-root users, central printer management, virtual filesystem plugins and others.

www.oreilly.com

Please send information about releases of Linux-related products to James Gray at newproducts@linuxjournal.com or New Products c/o *Linux Journal*, 1752 NW Market Street, #200, Seattle, WA 98107. Submissions are edited for length and content.



Liferay's Liferay Portal

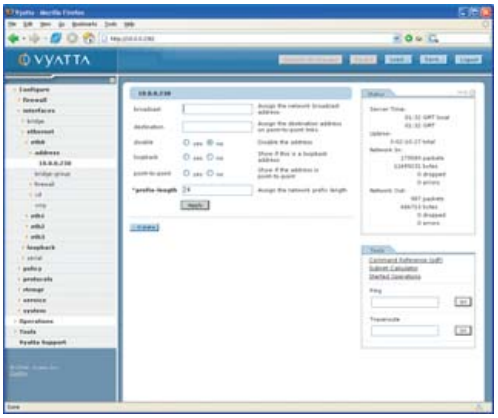
We've been given the lowdown on Liferay's new version 4.2 of the Liferay Portal, an enterprise-class, open-source portal framework. One of the key features in the updated edition involves integration with the ServiceMix Java Business Integration engine, which acts as a single point of connection for disparate enterprise applications, simplifying the integration, upgrade and substitution of siloed applications, such as CRM, ERP and ECM. Additional new features include a jBPM workflow engine, an Ajax-based chat messaging solution, Alfresco integration and a new parallel portlet rendering engine. Liferay is available for download from the company's Web site.

www.liferay.com

XenSource's XenEnterpriseT, XenServerT and XenExpressT

XenSource has gone hog wild with its concurrent announcement of three different virtualization products: XenEnterpriseT, XenServerT and XenExpressT. The products target the needs of "Fortune 50 enterprises, mid-market Windows standard server IT environments to technology enthusiasts and developers", respectively. All three products share a common, open-source, Xen-based architecture, which makes "the upgrade path seamless between products", sayeth XenSource. Xen allows multiple virtual server instances to run concurrently on the same physical server, with near native performance. XenEnterpriseT is designed for the heterogeneous environment of enterprises, delivering "bare metal" performance for virtualized guest operating systems. Meanwhile, XenServerT is intended for Windows standard server environments and "gives IT professionals a high-performance, easy-to-use virtualization platform for Windows". Finally, XenExpressT is made for for developers and technology enthusiasts, offering a "comprehensive, production-ready, free product, which enables anyone to get started with Xen virtualization quickly". All three products support Windows Server 2003/2000, as well as Windows XP, Red Hat Enterprise Linux, SUSE Linux Enterprise Server and Debian Sarge guests.

www.xensource.com



Vyatta's Open Flexible Router

Vyatta's Open Flexible Router (OFR), upgraded recently to version 1.1, is an open-source LAN and WAN routing solution for small- to medium-sized businesses. OFR's product benefits include not only all standard routing protocols and high-availability and security features, but also the ability to customize the product and add features as needed. The latter feature gives users flexibility in managing future requirements on their own terms rather than relying on the actions of closed-source vendors. Vyatta also states that OFR allows one to deploy an "enterprise-class router for a fraction of the cost of a traditional closed-source, proprietary router". This latest release adds support for T3 WAN connections and provides compatibility with third-party application packages from Debian GNU/Linux. Debian package compatibility extends OFR's capabilities to include third-party applications in the Debian universe, such as Openswan, Asterisk and ClamAV, among others. The OFR software is available for download from Vyatta's Web site.

www.vyatta.com

Storix's System Backup Administrator

Storix has brought forth a new version (6.0) of its flagship backup and disaster recovery application, System Backup Administrator (SBAdmin). SBAdmin not only backs up data files to a network server, but it also provides the ability to rebuild a system from the ground up on different hardware environments. One can reconfigure the system to restore onto completely different hardware, change filesystem types, migrate from disk partitions to LVM or software RAID and so on. Examples of new features are a Web-based interface, Oracle DB backup, remote system installation and balancing of the CPU load against the backup size. Supported OSes include Linux and AIX; supported hardware includes all Intel-based (x86/Pentium), IBM pSeries (32/64-bit) and HP Integrity (Itanium 2).

www.storix.com

Coyote Point Equalizer E550si Load Balancer

Load balancers help make a Web site mission-critical.

LOGAN G. HARBAUGH

Providing fault tolerance as well as the ability to scale beyond the capacity of a single server, load balancers are practically a necessity for any commercial site. Because loads on a given Web site can fluctuate by several orders of magnitude (five or six, in the case of sites like Victoria's Secret or the World Cup Soccer site), and given that thousands of dollars a minute may be lost if the site is unavailable, being able to spread the load across many servers and ensure that users can still connect, even if one or more physical servers fails or stops responding, is crucial.

Figure 1. Coyote Point Equalizer E550si Load Balancer

The latest load balancer available from Coyote Point Systems is the Equalizer E550si, a 1u (1.75"-high) appliance that offers 20 10/100/1000 ports, all the load-balancing features necessary to set up a sophisticated Web farm or other type of virtual cluster, and excellent performance, at a cost of $10,995 US.

You may be asking yourself, "Why do I need a load balancer?" Or, "Why should I pay that much for something I can get for free?" In its simplest form, load balancing simply distributes requests as they come in to one of several back-end servers in a virtual cluster, sharing the load equally among all the servers in a round-robin scheme. A DNS server can do this by mapping several IP addresses to the same host name, for instance:

```
www.store.com    192.168.0.10
www.store.com    192.168.0.11
www.store.com    192.168.0.11
```

The problem with using a DNS server in this fashion is that requests are distributed to each server in turn, whether or not that server is actually available, and regardless of how heavily loaded each of the servers is. Also, the first address in the list may be cached more often across the Web, resulting in higher loads on that server. Finally, many applications, such as e-commerce, can break unless a client is connected to the same server through its session, and there's no way to ensure this with a DNS round-robin setup.

Apache and Tomcat also can balance loads across a cluster of Apache and Tomcat servers, using a specialized Tomcat Worker instance. This type of load balancing is somewhat more sophisticated, allowing for checks to ensure that a host is available and adding more sophisticated algorithms than simple round-robin—for instance, allowing new requests to be sent to the least heavily loaded server. This type of load balancing can enable persistent sessions, so that a client can be directed to the same server for the duration of the session. However, this method will not work with other Web servers and will take some fairly specialized knowledge to set up and maintain.

A single-processor server with two standard NICs can't hope to match the millions of concurrent users and the levels of traffic that the Equalizer can, with a carefully tuned OS and 20 gigabit ports available.

There also are open-source load balancers, such as Ultra Monkey, which can offer sophisticated load-balancing algorithms, persistent sessions, health checking, failover to a backup load balancer and more. These can be installed on any Linux server and simply need one or two NICs to begin creating a virtual cluster.

So, why buy a $10,995 box when you can set up a server for a few hundred?

First, performance. A single-processor server with two standard NICs can't hope to match the millions of concurrent users and the levels of traffic that the Equalizer can, with a carefully tuned OS and 20 gigabit ports available.

Second, ease of use. The Equalizer comes with a very simple and straightforward Web-based GUI that any network admin can use to create an enterprise-class load-balanced cluster.

Third, the Equalizer can be used with any IP-based application, not only HTTP/HTTPS. It supports DNS, WAP, RADIUS, SMTP, POP, IMAP, NNTP, FTP and streaming media, as well as most other UDP- and TCP/IP-based protocols. It also can handle Active Server Pages, as well as Java application servers, and pretty much any kind of SQL back-end database server.

The Equalizer also offers an optional SSL acceleration card that provides SSL encod-

ing/decoding, which can reduce server loads quite substantially, and multiple Equalizers can be networked together to provide geographic load balancing, which allows you to set up several geographically separate Web sites that all serve the same URL, so that even if an entire data center is off-line, the others can continue to service users. The geographic load-balancing software, Envoy, can determine which data center will be able to respond the fastest to any given clients and to send those clients to the site that will give them the best service.

Setting up the Equalizer is a simple matter of performing the initial network configuration via serial terminal, then logging in to the system via the browser interface to configure one or more virtual clusters. Setting up a cluster is easily done by filling in the IP addresses of the servers in the cluster and making a few selections from drop-down boxes.

The major choices are the method of load balancing and the type of cluster. The load-balancing options are round-robin, static weight (set percentages of the total number of connections given to each server), adaptive, fastest response, least connections or server agent. Adaptive should be the default in most cases, as it combines the fastest response and least connections to provide very even server loads under most conditions. The type of cluster can be HTTP, HTTPS or any designated TCP/IP port range desired. Once a cluster is set up, you can be as granular as you like about creating persistent sessions, logging, reporting, monitoring services and servers to ensure availability, error handling or even automatically adding additional servers to a cluster as load increases. The default settings generally will be the optimal ones, but your ability to customize things is limited only by your ability to script actions.

For example, you can ping a server to ensure hardware connectivity, but you also can send a query via any text-based request/response protocol—not merely HTTP, but something like a Telnet-based SQL command—and verify that the response is valid. This means you can ensure that specific services are available on each member of a cluster, rather than just confirming that the network interface is operational. You can route traffic to a cluster based on rules that are written in standard POSIX.2 expressions. You could specify a rule that directs all traffic coming from a specific set of IP addresses to one cluster, and all other traffic to another, or match IP ranges assigned to specific countries to localize a Web site in other languages.

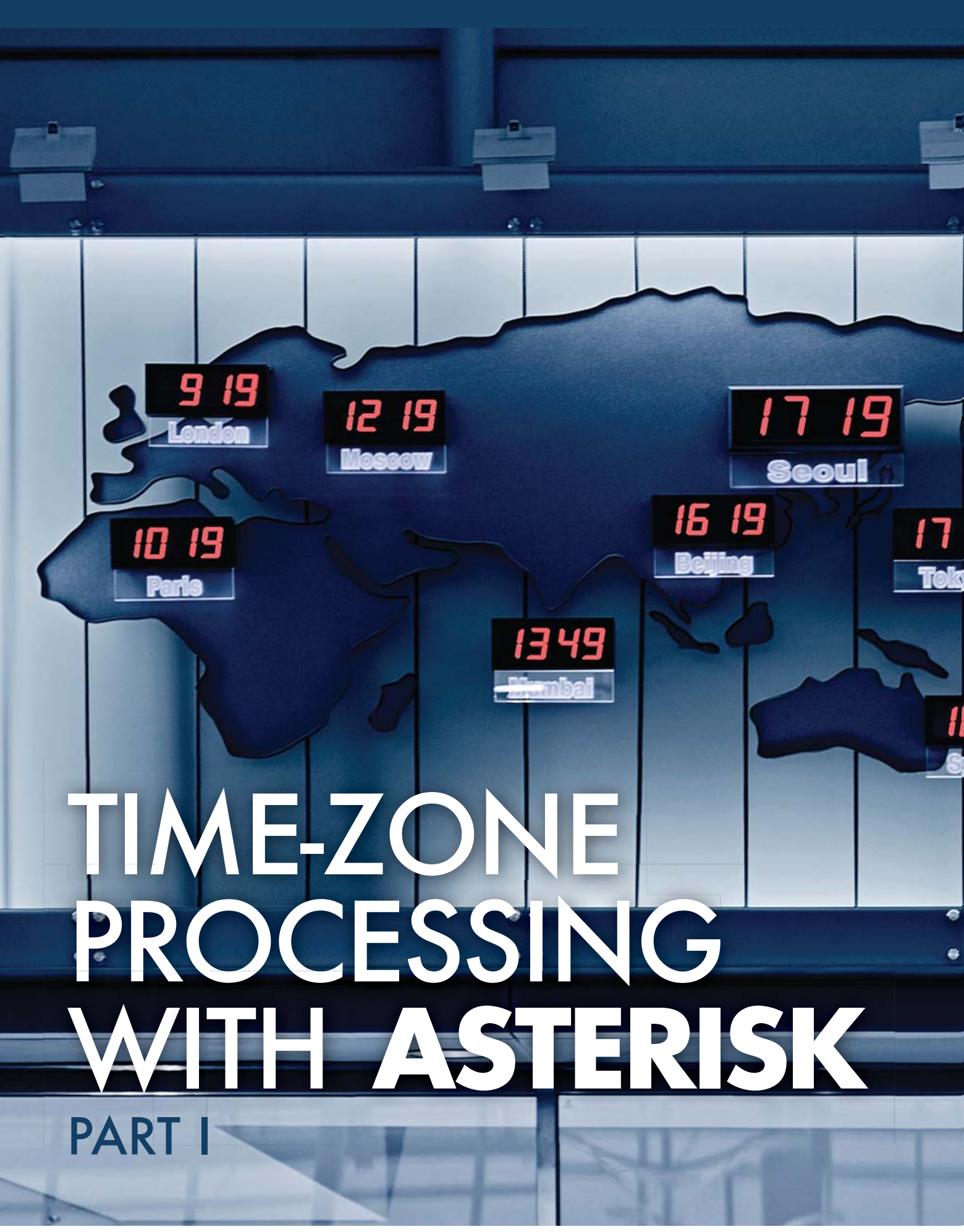
The Equalizer can automatically place cookies in the HTTP stream returned to a client so that it can identify a specific client and ensure that all traffic for that session comes to the same server. In addition, you can run scripts when a condition is met. For instance, you could define a rule that sends an e-mail if average loads on the cluster exceed 70% or even add additional servers to a cluster when loads are high.

Although there are load-balancing solutions that are less expensive than the Equalizer E550si (and many that are more expensive), the mix of high performance, ease of use and programmability is hard to beat.■

Logan G. Harbaugh is a freelance reviewer and IT consultant located in Redding, California. He has been working in IT for almost 20 years, and has written two books on networking, as well as articles for most of the major computer publications.

Coyote Point Equalizer E550si

- Coyote Point Systems: **www.coyotepoint.com**.
- Pricing starts at $10,995 US.
- Unlimited numbers of virtual clusters.
- Up to 128 servers per cluster.
- Bandwidth: 20Gb/s.
- Number of concurrent connections: approximately 10,000,000.
- Ports: 1–10/100, 20–10/100/1000.



TIME-ZONE PROCESSING WITH ASTERISK

PART I

Use Asterisk to adjust for your daylight hours.

MATTHEW GAST

Last year, I took a trip to Asia. To stay in touch, I carried a GSM world phone, capable of receiving telephone calls in the countries I was visiting. The capability to receive calls with the same mobile phone number I use at home while halfway across the world seemed incredibly cool—at least until the first call came in! Mobile phones hide the location of the phone, which cuts both ways. A colleague had decided to call me in the middle of the day on a Friday, which had awakened me very early on Saturday morning, because the phone "hid" my faraway location from him.

After returning home, I asked several people why my phone company could not simply play a message to warn callers when my time zone changes by more than four or five hours, letting them know the call might be inconvenient. Nobody could come up with a technical reason, but we all suspected it was because the mobile phone company to which I subscribed charged several dollars per minute to connect calls. As part of the process of attaching a GSM phone to a network, the home network needs to learn where the phone is visiting, and that information conceivably could include a time zone.

I returned to my idea once I started using Asterisk, because it provides an extensive toolkit for designing PBX-hosted services. Anything that can be coded in a computer can become an Asterisk service. After I understood the basics of Asterisk, I sat down to implement a feature that kept track of the time of day where I visited and prevented calls from coming in at inconvenient times.

The system I built on top of Asterisk to handle this feature has two major parts. The key to the system is maintaining a time-zone offset from the time in London. (My code implements offsets only of whole hours, though it could be extended to use either half or quarter hours.) When a device first connects to Asterisk, its IP address is used to guess the location and, therefore, the time offset. After the offset is programmed into the system, incoming calls are then checked against the time at the remote location. Before the phone is allowed to ring, the time at the remote location is checked, and callers can be warned if they are trying to complete a call at an inconvenient time.

STEP 1 ESTIMATING THE TIME ZONE

Asterisk does not have a built-in method to estimate the time zone from an IP address, but it does have the next best thing—the Asterisk Gateway Interface (AGI). AGI programming allows an Asterisk extension to pass data to an external program, do computations in that program and return results as Asterisk channel variables.

I began the project by writing an AGI script that would take an IP address as input and return an estimated time zone. Several existing geolocation databases map IP addresses into geographic information. None of the free products or compilations I tried for this project could return a time zone directly from the database, so I estimated the time zone based on the longitude. (The earth's surface has 24 time zones, each of which is approximately 15 degrees of longitude.) After trying several databases, I settled on MaxMind's GeoLite City, a free-for-non-commercial-use version of the comprehensive GeoCity commercial database.

GeoCity has APIs available in several programming languages. I used Perl, because there is also an AGI library module for Perl that makes handling AGI scripts easier. As input, the program takes an IP address, and then it needs to return the time zone. I used the convention of returning the time zone as an offset of the number of hours from the time in London, which leads to some differences in daylight time handling.

The start of the program pulls in utility functions, including the Asterisk::AGI module, which decodes all the parameters passed to the program by Asterisk:

```
#!/usr/bin/perl -w
# Asterisk AGI to estimate time zone from IP address

use strict;
use Asterisk::AGI;
use Geo::IP::PurePerl;
use POSIX qw(ceil floor);

# California is GMT -8
my $HOME_OFFSET = -8;

my $AGI = new Asterisk::AGI;
my %input = $AGI->ReadParse();
```

(The full listing of programs from this article is available on the *Linux Journal* FTP site; see Resources.) The argument to the program is an IP address, which is given to us by Asterisk. The first check is to determine whether the IP address is on the local subnet of the Asterisk server, 192.168.1.0/24. Most location databases don't include RFC 1918 address space and won't return a lookup. The MaxMind API can accept a domain name as an argument, but we don't expect to pass it one:

```
my $addr = $ARGV[0];
my @octets = split(/\./,$addr);

if (($octets[0] eq "192") && ($octets[1] eq "168") &&
   ($octets[2] eq "1")) {
  # Local IP addresses get the home offset
  $AGI->set_variable("TZ_OFFSET",$HOME_OFFSET);
  exit 0;
}
```

The use of the database is straightforward. We create a new object, telling the API the location of the database on disk, and then call the get_city_record_as_hash function in MaxMind's API, which returns everything about the IP address as a hash. The item of interest is the longitude component of the hash. If there isn't one, we'll simply return -8 for California and let Asterisk deal with the problem:

```
my $gi = Geo::IP::PurePerl-> new(
   "/usr/local/share/GeoIP/GeoLiteCity.dat",
   GEOIP_STANDARD);

my $cityref = $gi->get_city_record_as_hash($addr);

if (!(defined ($cityref->{"longitude"}))) {
  # Guess at the home time when longitude undefined
  $AGI->set_variable("TZ_OFFSET",-8);
  exit 1;
}

my $longitude=$cityref->{"longitude"};
```

A bit of math is required to deal with the fact that time-zone boundaries may be 15 degrees, but zero degrees is in the middle of a time zone. We can use two formulas, depending on whether the longitude is positive or negative. After computing the time zone, we pass it back to Asterisk in the TZ_OFFSET channel variable, where it is available for use in the dial plan:

```
my $quotient;
my $numerator;
my $denominator=15;

if ($longitude>0) {
   $numerator=$longitude+7.5;
   $quotient=floor($numerator/$denominator);
} else {
   $numerator=$longitude-7.5;
   $quotient=ceil($numerator/$denominator);
}

$AGI->set_variable("TZ_OFFSET",$quotient);
```

STEP 2
CONFIRMING THE TIME-ZONE INFORMATION

As convenient as it would be to have a reliable database of IP addresses mapped into the right time zones, there is still the problem of handling summer shifts in time. Plus, the estimate comes from a demonstration database that is not guaranteed to be accurate. Therefore, the AGI script is called from within an extension that is used to confirm the estimate or save a new one. For the confirmation step, I created an extension with the number *89 (because 8–9 is the numerical mapping of the letters T–Z). Just as with the previous program, some of the debugging statements are removed for brevity, but the full version is available from the *LJ* FTP site (see Resources).

This article shows the dial plan information entry in the Asterisk Expression Language (AEL). I started using AEL because it has better control structures and it is much easier to write structured code. For voice menus, the superior control structures in AEL allow much easier validation of input.

A call into the extension begins with a welcome announcement and a call to the script (shown previously) to estimate the time zone. All of the time-zone confirmation greetings are stored in the msg/tz subdirectory of the Asterisk sound library. Asterisk's SIPCHANINFO can be used to get SIP channel information. Specifically, the value of SIPCHANINFO(recvip) is the routable Internet address that the remote device used to register and, therefore, will work even if the remote device is behind a network address translator:

```
_*89 => {
      Answer;
      Playback(msg/tz/tz-wizard-welcome);
      Set(PEERIP=${SIPCHANINFO(recvip)});

      // Geolocate as a first stab at time zone
      AGI(tz-lookup.pl,${PEERIP});
      NoOp(TZ offset from script is ${TZ_OFFSET});
```

The script returns a guess at the time zone of the IP address used by the SIP peer in the TZ_OFFSET variable. However, the whole purpose of this call is to confirm the offset, so we proceed to a series of confirmation steps. The system starts by giving a confirmation of the time offset from the time in London and uses that to say the time. Asterisk keeps internal time as epoch time (the number of seconds past midnight on January 1, 1970 GMT) and converts it to local time for a given time zone. I am handling any adjustments to the time for summer months with a sledgehammer, which assumes that most of the countries I visit will be on roughly the same schedule as the UK and corrects any errors later in the validation step:

```
playoffset:
   Playback(msg/tz/tz-you-are-at);
   SayDigits(${TZ_OFFSET});
   Playback(msg/tz/tz-hours-to-london);

playtime:
   // London time keeps summer time
   Playback(msg/tz/tz-current-time-is);
   SayUnixTime($[${EPOCH}+${TZ_OFFSET}*60*60],
```

```
        Europe/London,A \'digits/at\' IMp);
```

Next, we ask the user to confirm whether the time is correct. The Read() application gets one digit from the user. AEL's switch statement is very handy for working with user input, because it can be used to set up a series of actions for a voice menu quite easily without extensive use of a forest of branching statements. In this case, the switch statement offers the option of a one-hour correction by entering the number 1, an arbitrary correction with the number 2 and an error statement that jumps back to the start of the time readout if anything else is pressed. The only catch in using goto within a switch statement in AEL is that due to the internal representation of the control structures, you must use a fully qualified goto, including the context, extension number and label. My internal extension context is from-internal, so gotos are prefaced with `from-internal|*89`:

```
Read(INPUT,msg/tz/tz-confirm-correct,1);
switch (${INPUT}) {
     case 1:
          goto from-internal|*89|expiry;
     case 2:
          goto from-internal|*89|correction;
     default:
          Playback(msg/tz/tz-1-or-2-please);
          goto from-internal|*89|playoffset;
};
```

At the correction label, there is a second option. I expect that summer-time errors will be common in actual use, so I added an expedited one-hour correction that can add or subtract an hour easily. The menu that handles the type of correction and the one-hour correction subroutine is available on the *LJ* FTP site. In structure, both are similar to the switch statement shown previously.

When I arrive in faraway locations, I am usually quite tired, and my ability to do mental arithmetic is substantially reduced from my capacity when fully functional. Rather than a voice menu to select a location, the menu prompts for the local clock time and computes the time-zone offset from the time in London. The basic algorithm is to obtain the 24-hour reference time in London with Asterisk's STRFTIME function and compute the offset from the time entered by the user. There's a possibility that the resulting offset will be too big or too small, so the script corrects for that:

```
gmtskew:
   Read(INPUT,msg/tz/tz-24-hour-prompt,4);
   Set(REMOTEHR=${INPUT:0:$[${LEN(${INPUT})}-2]});
   Set(REFERENCEHR=${STRFTIME(${EPOCH},
        Europe/London,%H)});
   Set(TZ_OFFSET=$[${REMOTEHR}-${REFERENCEHR}]);
   // correct for too big/too small offsets
   if ( ${TZ_OFFSET} > 12 ) {
        Set(TZ_OFFSET=$[${TZ_OFFSET}-24] );
   };
   if ( ${TZ_OFFSET} < -12 ) {
        Set(TZ_OFFSET=$[${TZ_OFFSET}+24] );
   };
   Return;
```

When the user confirms the time offset, the changes are saved in the Asterisk database with the DB function. Part of saving the changes is to ask the user for how long the offset should be saved. This code prompts for the number of days, though that part of the code easily can be extended to ask for an expiration date and time. After determining the time when the offset expires, the offset is read back to the caller in both the time at the remote site, as well as the home location. (Note that my home location is US/Pacific; you will need to replace that with your own time zone.)

For record-keeping purposes, four variables are stored with the name of the extension that controls the offset. There is the actual offset, as well as a start time, expiration time and the IP address of the SIP peer. If the device is moved before the end of the offset, we want to re-confirm the time offset automatically:

```
expiry:
     Set(NOW=${EPOCH});
     Set(CURRENT_OFFSET_TIME=$[ ${EPOCH} +
          ${TZ_OFFSET}*60*60 ]);
     Set(DB(tz/${PEERNAME}-TIMESKEW)=${TZ_OFFSET});
     Set(DB(tz/${PEERNAME}-TIMESKEW_START)=${NOW});
     Set(DB(tz/${PEERNAME}-TIMESKEW_ADDR)=${PEERIP});

     Playback(msg/tz/tz-your-offset-of);
     SayDigits(${TZ_OFFSET});
     Playback(msg/tz/tz-hours-relative-to-london);
     Playback(msg/tz/tz-confirm-time);
     SayUnixTime(${CURRENT_OFFSET_TIME},
          Europe/London,A \'digits/at\' IMp);

expiration-confirm:
     Read(TZ_DURATION,msg/tz/tz-days-active,2);
expiration-readout:
     Set(DB(tz/${PEERNAME}-TIMESKEW_END)=
          $[${NOW}+24*60*60*${TZ_DURATION}]);
     Playback(msg/tz/tz-shift-active-for);
     SayDigits(${TZ_DURATION});
     Playback(msg/tz/tz-days);
     Read(INPUT,msg/tz/1-if-right--2-if-wrong,1);
     switch (${INPUT}) {
          case 1:
               // Everything is OK, read out results
               NoOp(Go to result read-out);
               break;
          case 2:
               goto *89|expiration-confirm;
          default:
               Playback(msg/tz/tz-1-or-2-please);
               goto *89|expiration-readout;
     };
```

AEL's switch statement is very handy for working with user input, because it can be used to set up a series of actions for a voice menu quite easily without extensive use of a forest of branching statements.

STEP 3 LETTING CALLERS KNOW THE TIME

At this point, Asterisk has all the data it needs to restrict calls based on the time of day. The Asterisk dial plan can be used to check the time at the remote site; if it is before, say, 8 am or after 10 pm, the phone plays the remote time to the caller and asks whether the extension should ring anyway.

First, Asterisk needs to pick up the phone and compare the current local time to the 8 am to 10 pm window. Asterisk's STRFTIME function converts an epoch into a time of day. By adjusting the current epoch time with the offset value, the STRFTIME function returns a time of day. A call that is too early or too late jumps to code that silences the ringer. When a call's ring is silenced, this example code allows the caller an override:

```
300 => {
      Answer;
      Playback(msg/remote-extension-greeting);
      Set(TZ_OFFSET=${DB(tz/${EXTEN}-TIMESKEW)});
      Set(RMT_EPOCH=$[${EPOCH}+${TZ_OFFSET}*60*60]);
      Set(REMOTE_CLOCK=${STRFTIME(${RMT_EEPOCH},
            Europe/London,%H:%M)});

      if("${REMOTE_CLOCK}" < "08:00") {
            goto 300|too-early;
      };
      if("${REMOTE_CLOCK}" >= "22:00") {
            goto 300|too-late;
      };

      goto 300|normal-ring;
```

The confirmation code starts by silencing the ring, but allows the user to enable ringing by pressing one. I've allowed any caller the flexibility to override my hours, but this conceivably could be handled by allowing only select callers to be able to override. The "standard" way to set the ring tone is to set the ALERT_INFO channel variable. My ATA is a Sipura, which allows the definition of eight ring cadences. I have defined a cadence called silence, which is a ring that never uses the bell:

```
too-early:
     NoOp(Too early to ring);
too-late:
     NoOp(Too late to ring);
     Set(__ALERT_INFO=silence);
override-silence:
     Playback(msg/my-remote-time-is);
     SayUnixTime(${RMT_EPOCH},Europe/London,
          A \'digits/at\' IMp);
     Read(CONFIRM,msg/press-1-to-confirm-call,1);
     if (${CONFIRM}=1) {
          Set(__ALERT_INFO=Bellcore-r1);
     };
```

Ringing the phone is straightforward, because it requires using only the Dial application, and the ring cadence has been set elsewhere. Alternatively, the call can be sent straight to voice mail:

```
normal-ring:
     Dial(SIP/300,20);
vm-only:
     VoiceMail(umatthew);
Hangup;
};
```

For enhanced code re-use, the time-of-day checks could be incorporated into a macro that is called as part of every extension.

USING THIS PROJECT

As a first step, install the GeoLiteCity database from MaxMind, and install the time-zone lookup script into /var/lib/asterisk/agi-bin. To call that script, add the time-zone configuration extension *89 to your dial plan. Every extension that requires time-of-day-specific treatment needs to have its dial plan modified with code similar to that shown in Step 3 of this article.

Then, as a user of the PBX, every time a SIP extension is registered, you need to call *89 to set up the time zone. This need to initiate the configuration process manually is somewhat annoying. Asterisk does provide an interface that can be used to set up the call to the user automatically, which I will describe in a follow-up article.

On a PBX that supports multiple users, several items would benefit from centralized storage. In this example, the time-of-day comparison is coded into the target extension's dial plan. By storing this data in the Asterisk database, it would be possible to let users change their time-of-day schedules without administrator intervention.

Finally, I also have configured my PBX with a "friends and family" override feature that allows selected callers to complete a call even if it normally would be blocked. Callers on the privileged list can request the time of day at my location and are allowed to ring a call through even if it would ordinarily be blocked.■

Matthew Gast is the author of the leading technical book on wireless LANs, *802.11 Wireless Networks: The Definitive Guide* (O'Reilly Media). He can be reached at matthew.gast@gmail.com, but only when he is close to sea level.

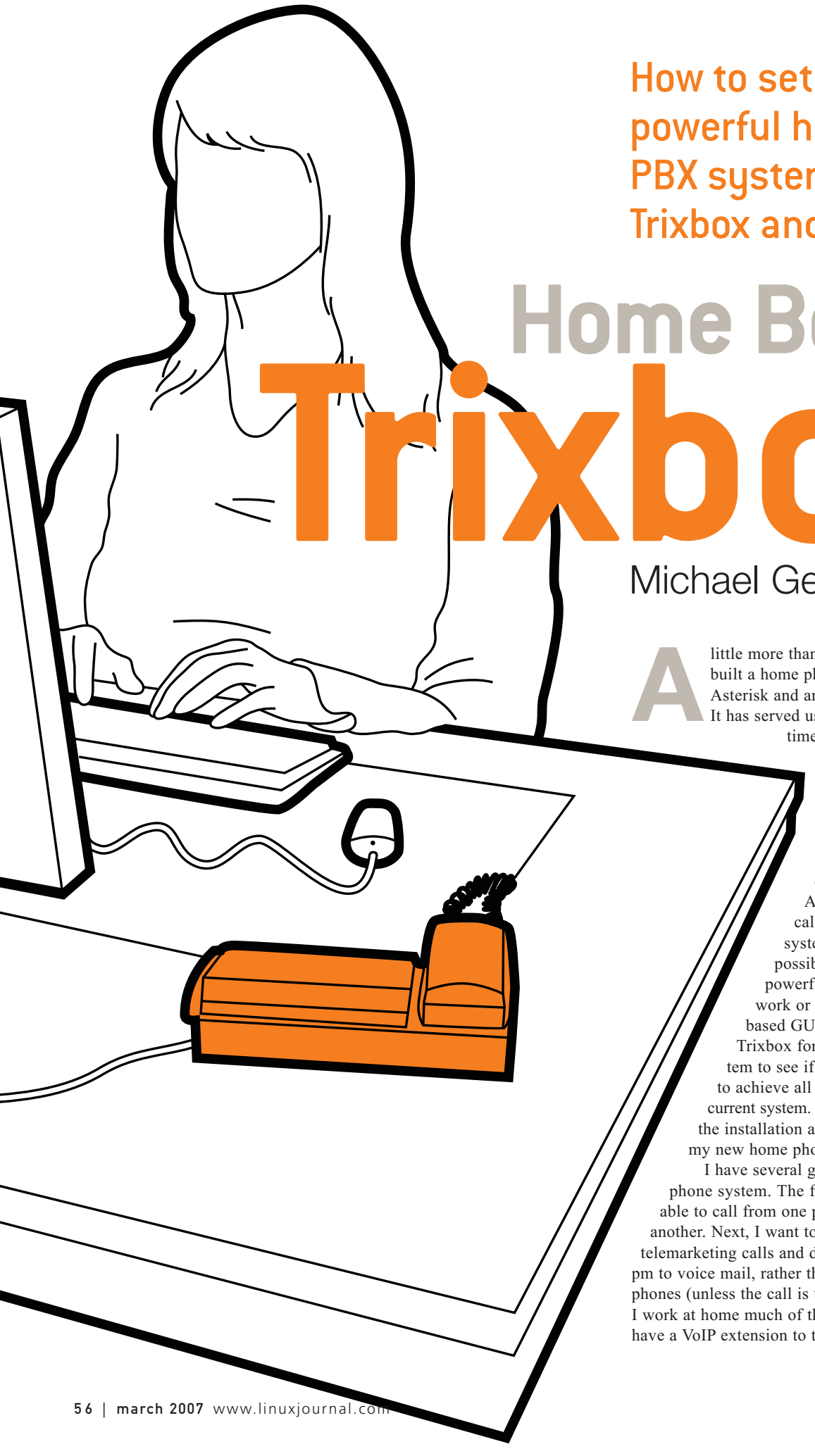
Resources

Full source listings of all files:
ftp.linuxjournal.com/pub/lj/issue155/9190.tgz (The dial plan code for the time-zone confirmation menu is 9190l1.txt, the dial plan code for an extension that is time-zone-aware is 9190l2.txt and the AGI script for time-zone geolocation lookup is 9190l3.txt.)

MaxMind GeoLite City Database (Download and API):
www.maxmind.com/app/geolitecity

Asterisk: **www.asterisk.org**

Asterisk AEL: **www.voip-info.org/wiki/view/Asterisk+AEL**



How to set up a powerful home PBX system with Trixbox and FreePBX.

Home Box to Trixbox

Michael George

A little more than two years ago, I built a home phone system using Asterisk and analog interface cards. It has served us quite well, but it is time for an upgrade. In the time since first constructing that system, great strides have been taken in simplifying Asterisk configuration. Asterisk@Home (now called Trixbox) is a system that makes it possible to implement a powerful phone system for work or home using a Web-based GUI. I decided to use Trixbox for my new phone system to see if it has the flexibility to achieve all the features of my current system. This article discusses the installation and configuration of my new home phone system.

I have several goals for the new phone system. The first goal is to be able to call from one part of the house to another. Next, I want to minimize annoying telemarketing calls and direct calls after 10 pm to voice mail, rather than ringing the phones (unless the call is urgent). Because I work at home much of the time, I want to have a VoIP extension to the office system.

Assembling and Installing the Phone System

Before configuring the phone system, you must assemble it and install the software. Although Trixbox makes this rather easy, I outline the installation process here, so that those who want to try it for themselves will know what to expect.

The first things to consider are the hardware and software pieces needed for this project. I already have an old spare computer to use (a Pentium II 266MHz with 256MB of RAM and 6GB HDD), but I also need an interface card for the phone lines. When I first built my current system, I got a Digium TDM31B with three ports for phone sets (FXS) and one telephone company (FXO) port, which now costs around $350 US. Purchasing from Digium helps support Asterisk, because it is the primary supporter of the project. The last requirement is the system software, Trixbox, which can be downloaded from **www.trixbox.org**.

The Digium card came configured with ports 1–3 for the phone sets and port 4 for the incoming phone line. Because managing phone interfaces has real-time demands, it is best to keep the Digium device on its own interrupt, especially in older machines like mine. Isolating the card on its own interrupt is usually not very difficult—changing slots and disabling unnecessary devices in the BIOS usually will do the trick. It is worth noting that because of the high interrupt rate of these interface cards, one should not run many other devices or applications with high interrupt frequencies, especially not X11.

Installing the Trixbox Software

After downloading the Trixbox ISO, I burned it onto a CD and booted the system from it. I was presented with a screen that told me that continuing will completely destroy any data on the hard drive. Willing to do that, I pressed Return to continue. The install process asked me to select a keyboard type and layout, a time zone and a root password. There seemed to be some very long pauses between these selections, but the system completed the install and configured the bootloader. At that point, with the base system installed and the network configured, it rebooted.

After coming back up, it retrieved the Trixbox software itself, built and installed everything and initiated another reboot. At reboot, it finished the installation of FreePBX and restarted. These reboots were a bit disconcerting at first, but by this point, the entire system was fully installed and running. However, it did not recognize or configure my TDM card.

One of the main features of Trixbox is its upgradability, so before panicking about the TDM configuration, I decided to update first. After logging in as root, I ran `trixbox-update.sh`. Twice. The first run updated the script itself and the second updated the rest of the system. After finishing, it told me to reboot the system. That time, all the TDM-related modules loaded, though only wctdm and zaptel were needed. The card was detected correctly, and running `genzaptelconf` from the command line configured it correctly. At this point, the Trixbox system was installed, up to date and functional.

Networking is configured easily from the command line with the `netconfig` command. I could leave it with a dynamic IP address, as I have no VoIP phones to connect, but my intention was to remove the keyboard and the monitor from the system and administer it remotely. Doing that is much easier if I know the internal network address of the system, so I set it to a static address and configured the rest of the network settings appropriately. I also changed the hostname of the system by modifying /etc/hosts and /etc/sysconfig/network and rebooting.

After reboot, I opened a browser window and directed it to the static address I defined. I then saw the initial Trixbox screen, as shown in Figure 1. Most configuration is done through FreePBX, which is found by selecting System Administration, entering the maint user name and the password from above (Figure 2) into the pop-up window and then selecting FreePBX. The FreePBX is very modularized, and those modules still needed to be installed and activated before I could

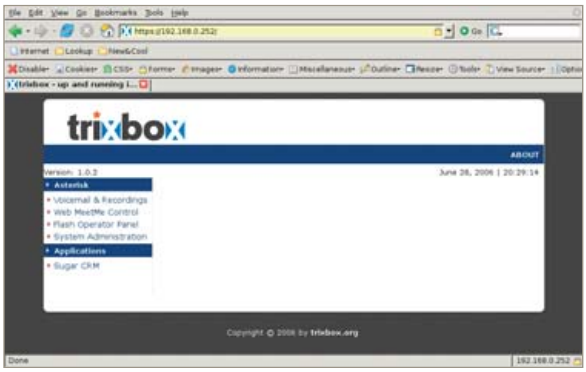


Figure 1. Trixbox Web Page Up and Running

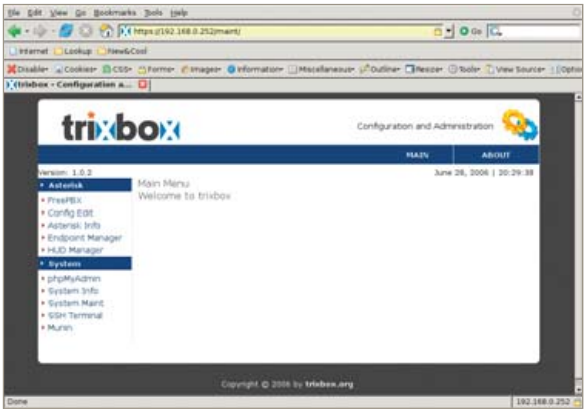


Figure 2. Main Menu after Logging In

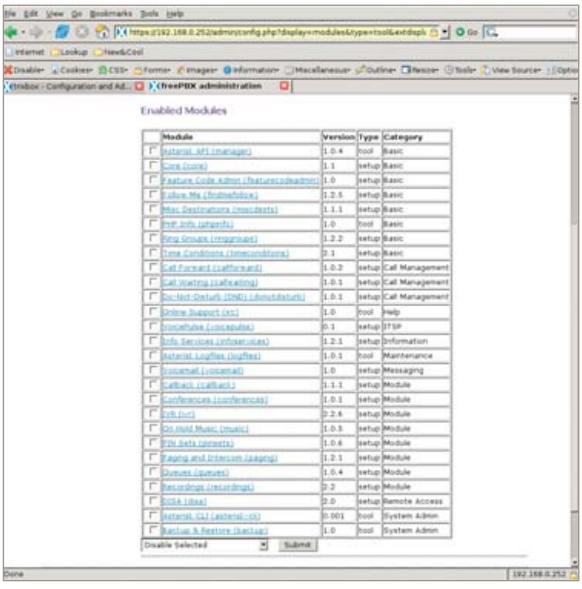


Figure 3. Module Setup in Trixbox

Figure 4. The FreePBX Browser Window

Figure 5. General Settings for FreePBX

Figure 6. Using the FreePBX Interface to Add Trunks

Figure 7. Edit the ZAP Trunk

Figure 8. Edit the IAX2 Trunk

Figure 9. Configuration of the IAX2 Trunk

do anything with the system. By first selecting Tools from the top menu and then Module Admin from the side, I was at the module management screen. I then selected the Connect to Online Module Repository link to view all the modules available. I selected all the Local modules and clicked the Submit button at the bottom of the table. Next, I downloaded and installed all the Online Modules except for Gabcast and Java SSH.

The final set of modules, shown in Figure 3, is a very comprehensive set of functionality for a small home system. This module management interface also checks to see if any module updates are available whenever connecting to the on-line repository. Those updates can be downloaded and installed easily to keep the system current.

Configuring the Phone System

After I had all the necessary software downloaded, updated, installed and configured, the phone system had all of its modules up to date and was ready for configuration. From the FreePBX browser window, I selected Setup from the top menu (Figure 4). Down the left are the menu items associated with many of the modules that were loaded. First, I changed the General Settings by selecting that menu item. One very useful feature of the FreePBX interface is that many of the items on the screens have pop-up windows associated with them to provide information for those items. I like to drop the r from the Dial command options to generate ringing tones to the caller only "when appropriate" and add w to the Dial command options and W to the Outbound Dial command options to allow our internal extensions to record calls in either direction by pressing *1 during the call. I also like to be able to transfer outbound calls between extensions if desired, so I add T to the outbound options. Finally, I changed the ring time to 20 seconds to give us more time to answer a phone. Figure 5 shows the final state of our General Settings.

Adding Trunks to the System

The next addition to the system was Trunks, which are where calls come in to or go out of the system. Selecting Trunks from the left-side menu displays the Add a Trunk screen (Figure 6). The trunk ZAP/g0 is created at installation, and it refers to all the sockets on the Digium interface card that connect to the phone company. In my case, that is only socket 4. I did not modify the default configuration of that trunk (Figure 7). One could set the Outbound Caller ID, but I leave that for the phone carrier to set. This trunk will be used for most calls through our system.

Another trunk I defined is an IAX2 trunk that connects to the office PBX, so I can receive calls sent to my work extension and make calls through the office account. Starting from the Add a Trunk screen (Figure 6), I selected Add IAX2 Trunk and filled in the configuration page for the trunk. Figures 8 and 9 show the configuration of that trunk. The PEER Details, USER Details and Register String have been changed to remove the IP address and

actual size

- Lowest cost smart switch
- Full featured routing
- Five independent Ethernet ports
- MDI-X 10/100 support
- RouterOS Firewall/Routing
- Passive PoE up to 120m (390ft)
- Input voltage 9-30V
- 175MHz MIPS CPU
- 64MB NAND and 32MB SDRAM
- Serial console port
- 115mm x 90mm (4.53in x 3.54in)
- Aluminium black anodized case
- Board price only $69

RouterBOARD™ 100 Series

made by routerboard.com

Figure 10. Default device options are sufficient for ZAP extension 22.

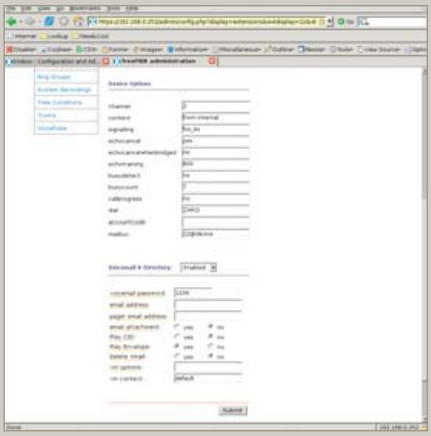


Figure 11. More Default Device Options

Figure 12. Similar Configuration for Kitchen Extension 24

Figure 13. More Default Device Options for Extension 24

Figure 14. The Schoolroom extension 23 leaves voice mail disabled.

Figure 15. The ring group 20 will ring the whole house.

passwords for the work system, but all these settings are present by default when the trunk is created. Usually only the address, user name and secret need to be changed. Two things to note here:

1. The USER Context setting should be the name of the account from the service provider.

2. I have changed the context for calls coming in on this trunk to the custom-from-work context, which I describe later in this article.

Once I filled in all the parameters, I selected the Submit Changes button to create the trunk. Then, as with all changes to the configuration, the interface displayed the red bar at the top of the screen for committing the changes to the system. With the trunks defined, there is now a way for calls to arrive in to and depart from the system.

Setting Up Internal Dialing

Next, I needed to define internal extensions for the Kitchen, Schoolroom and Office.

After selecting Extensions from the left-side menu, I chose to Add a ZAP extension. I gave extension number 22, display name Office and channel 2 to my office phone. Giving it channel 2 means it is plugged in to socket 2 on the Digium PCI card.

I then enabled voice mail on that channel and assigned the required voice-mail password. After selecting Submit, the new channel appears in the right-side menu. When it is selected again from the menu, the Device Options have more parameters that could be modified, but the defaults are sufficient (Figures 10 and 11). Extension 24, assigned to the Kitchen phone on channel 1, is configured similarly (Figures 12 and 13), and the voice mail on this extension is the family voice-mail box. The Schoolroom, assigned extension 23 and channel 3, is also similar but voice mail is left disabled (Figure 14). That completes all three of the real extensions on the system.

I also designed one pseudo-extension that will ring the whole house and another to ring only the Kitchen and Schoolroom phones. I used Ring Groups for this. A Ring Group is a set of extensions that are associated with each other, can be dialed with a single number and can ring in a specific order. Selecting Ring Groups from the left-side menu, I defined group number 20 for the whole house. I put all three real extensions into the extension list and set them to all ring at once, which is the strategy ringall (Figure 15).

I set the ring time to 20, the same timeout as the other extensions, and if no one answers, the call will go to the family voice-mail box (24). Making a ring group for the house extensions is exactly the same, but only extensions 23 and 24 are in the list (Figure 16). These ring groups can be dialed from any internal extension or from an incoming call that is allowed to

Polywell's Ultimate Linux Systems

More Choices, Better Service, Great Value

Tower Systems

4-way Opteron, OpenGL QFX PolyStation 2055A-2210

-2xAMD Opteron™ Dual-Core 2210 Processors
-2GB Dual Channel 667MHz DDR2 ECC
-NVIDIA Quadro™FX 1300 OpenGL Graphics
-2x250GB SATA Drives, On-board RAID-5
-Sony 16X DVD-RW, Logitech K/B, Mouse
-2x Gigabit LAN, 2 x PCIe 16x Slots
-Linux / BSD / Windows OS Supported

$1999 (custom config. available)

4-way Opteron, SAS Drives PolyStation 2055SS-2216

-2xAMD Opteron™ Dual-Core 2216 Processors
-8GB Dual Channel 667MHz DDR2 ECC
-NVIDIA Quadro™FX 3500 OpenGL Graphics
-2x15K RPM 73GB SAS Drives, SAS Controller
-Sony 16X DVD-RW, Logitech K/B, Mouse
-2x Gigabit LAN, 2 x PCIe 16x Slots
-Linux / BSD / Windows OS Supported

$4695 (custom config. available)

4-way Opteron, 32GB RAM Tower Server 2500M-2220

-2xAMD Opteron™ Dual-Core 2220 Processors
-32GB Dual Channel 667MHz DDR2 ECC
-11-Bay Tower with 2x700W Power Supply
-3TB 6x500GB SATA Drives, RAID-5
-Sony 16X DVD-RW, Logitech K/B, Mouse
-2x Gigabit LAN, 2 x PCIe 16x Slots
-Linux / BSD / Windows OS Supported

$9500 (custom config. available)

Desktop Systems

Low-Cost Linux Appliance PC Poly 485Ax-2800SP

-AMD Sempron™ 2800+ 64-bit Processor
-256MB 667MHz DDR2 Memory
-ATI X1000 PCIe Graphics
-80GB SATA Drive, SATA-RAID Controller
-16X DVD-ROM Drive, Logitech K/B, Mouse
-100Mbit LAN, 2 PCI + 1 PCIe 16x Slots
-Linux / BSD / Windows OS Supported

$299 (for volume purchase only)

2TB Storage Appliance MiniBox 430AM2-3000SP

-AMD Athlon™64 3200+ 64-bit Processor
-512MB Dual-Channel 667MHz DDR2 Memory
-NVIDIA GeForce™ 6150 Graphics
-2TB 4x500GB SATA Drives, RAID-5
-Optional DVD or RW Drive, K/B, Mouse
-Gigabit LAN, 2 PCI + 1 PCIe 16x Slots
-Linux / BSD / Windows OS Supported

$1295 (custom config. available)

High-End PC, 4G DDR2 Poly 430AM2-FX62

-AMD Athlon™64 Dual-Core FX-62 Processor
-4GB Dual Channel 667MHz DDR2
-NVIDIA GeForce™ 7900GS Graphics
-10K RPM 74GB + 320GB SATA Drives
-16X DVD-RW, 20-in-1 Reader, 1394a
-Gigabit LAN, 2 PCI + 1 PCIe 16x Slots
-Linux / BSD / Windows OS Supported

$2250 (custom config. available)

Rack Servers

Low-Cost ISP RackServer 1102EV-17 485Ax-3000SP

-AMD Sempron™ 3000+ 64-bit Processor
-512MB 667MHz DDR2 Memory
-ATI X1000 PCIe Graphics
-80GB SATA Drive, SATA-RAID Controller
-1U 17" Short Rack, 1 Slim-CD, 2 x 3.5" Bays
-100Mbit LAN, 1 Optional PCIe or PCI RISER
-Linux / BSD / Windows OS Supported

$399 (for volume purchase only)

2GB RAM, 500GB ISP Server 1102EV-17 430AM2-4200

-AMD Athlon™64 X2 Dual-Core 4200+ Processor
-2GB Dual-Channel 667MHz DDR2 Memory
-NVIDIA GeForce™ 6150 Graphics
-500GB 2x250GB SATA Drives, RAID-5
-1U 17" Short Rack, 1 Slim-CD, 2 x 3.5" Bays
-Gigabit LAN, 1 Optional PCIe or PCI RISER
-Linux / BSD / Windows OS Supported

$999 (custom config. available)

2GB ECC, 1TB ISP Server 1102EV-17 1000SL-1210

-AMD Opteron™Dual-Core 1210 Processor
-4GB Dual Channel 667MHz DDR2 ECC
-Integrated Graphics, 2 x Gigabit LAN
-1TB 2x500GB SATA Drives, RAID-5
-1U 17" Short Rack, Slim-CD+2x3.5" Bay
-Optional PCIe or PCI RISER Slot
-Linux / BSD / Windows OS Supported

$1799 (custom config. available)

High-Density Multi-Processor Servers

16-way Opteron, 128GB RAM 5124AIS 8850S-8212

-8 x AMD Opteron™ Dual-Core 8212 Processors
-128GB Dual Channel 667MHz DDR2 ECC
-12TB 24x500GB SATA Drives, RAID-5
-4 x Gigabit Ethernet, 4 PCI-X, 2 PCIe Slots
-5U 24-Bay RackCase, Redundant Power Supply
-Linux / BSD / Windows OS Supported

16-way 8212, 128G RAM, 24x500G $99999
8-way 8212, 32GB RAM, 8x500GB $15999

18TB 4U Storage, 32GB RAM 4024AIS 2500M-2210

-2 x AMD Opteron™ Dual-Core 2210 Processors
-32GB Dual Channel 667MHz DDR2 ECC
-18TB 24x750GB SATA Drives, RAID-5
-2 x Gigabit Ethernet, 4 PCI-X, 2 PCIe Slots
-4U 24-Bay RackCase, Redundant Power Supply
-Linux / BSD / Windows OS Supported

18TB 24x750G, 32G, 2x2210 $19999
11TB 22x500G, 2GB, 1x2210 $9999

6TB 2U Storage, 16GB RAM 2012SAS 2500M-2210

-2 x AMD Opteron™ Dual-Core 2210 Processors
-16GB Dual Channel 667MHz DDR2 ECC
-6TB 12x500GB SATA Drives, RAID-5 Controller
-2 x Gigabit Ethernet, 4 PCI-X, 2 PCIe Slots
-2U 12-Bay RackCase, 3 RISER or 7 Lowprofile
-Linux / BSD / Windows OS Supported

6TB 12x500G, 16G, 2x2210 $7999
1.5TB 6x500G, 2GB, 1x2210 $3399

2TB 1U Server, 32GB RAM 1104SC 2500M-2210

-2xAMD Opteron™Dual-Core 2210 Processors
-32GB Dual Channel 667MHz DDR2 ECC
-6TB 4x500GB SATA Drives, RAID-5
-2 x Gigabit Ethernet, 1 PCI-X or PCIe RISER
-1U RackCase, 1 Slim CD, 4 Swap HD Bays
-Linux / BSD / Windows OS Supported

2TB 4x500G, 32G, 2x2210 $7888
1TB 4x250G, 2GB, 2x2210 $1999

AMD Dual-Core technology enables one platform to meet the needs of multi-tasking and multi-threaded environments; provides platform longevity

Polywell OEM Services, Your Virtual Manufacturer
Prototype Development with Linux/FreeBSD Support
Small Scale to Mass Production Manufacturing
Fulfillment, Shipping and RMA Repairs

AMD 64 Opteron

AMD 64 Athlon X2

- 20 Years of Customer Satisfaction
- 5-Year Warranty, Industry's Longest
- First Class Customer Service

888.765.9686
www.Polywell.com/us/LJ

Polywell Computers, Inc 1461 San Mateo Ave. South San Francisco, CA 94080 650.583.7222 Fax: 650.583.1974

POLY POLYWELL

Figure 16. Ringing the Whole House from an Extension

Figure 17. Handling 9 for Outside Lines and 911 Separately

Figure 18. Set up an outgoing recording from extension 22.

Figure 19. Set up options for recording a message.

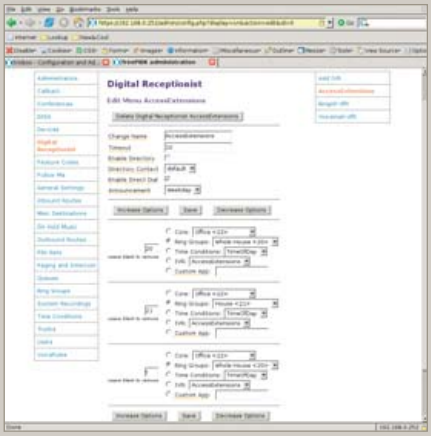


Figure 20. Set up the Digital Receptionist.

Figure 21. Set up ring-all defaults.

dial extensions directly. One nice thing about using these ring groups is that during the school year, I can drop the extension in the Schoolroom from the ring group, so people in that room are not disturbed during the day. In the summer, I can add that extension back into the group.

The system has three devices for receiving and placing calls, and they can be accessed by five extension numbers.

Calling the World

Although calling between extensions is certainly cool and useful, calling out of the system is still required. Outbound Routes is where that gap is bridged. For ease of configuration and for faster parsing of dialed numbers, I configured the system so that all numbers dialed out through the phone company will be prefixed with the digit 9. Selecting Outbound Routes from the left-side menu, I defined a route called 9_outside (Figure 17) for passing outgoing numbers to the PSTN trunk, which is ZAP/g0.

ZAP/g0 is a group containing all the ZAP channels that connect to the phone company, which is defined by default at installation by Trixbox. These groups are similar to the phone system concept of a hunt group: the channels are tried in sequence and the first one that is available will be used.

The dial patterns that I direct to this trunk are 911 and 9|.. The former pattern indicates that when 911 is dialed by an extension, the system dials 911 on the chosen trunk. The latter pattern (9|.) matches any dialed number that begins with a 9, followed by any number of digits.

The system then strips off the 9, as indicated by the vertical bar (|), and sends the remainder of the digits to the trunk. I handle 911 separately from other 9-prefixed numbers, because I do not want anyone to have to know or remember to prefix 911 with a 9 to get out (9911). Because there is only one trunk for dialing out and outgoing numbers must be prefixed with a 9, the outbound routes were very easy to define.

Handling Incoming Calls

The most complex part of the phone system is handling the incoming calls as designed. There are three parts to configure for incoming calls: audio messages to be played to the caller, Digital Receptionist menus, also known as Interactive Voice Response (IVR) menus, and the Time Conditions that determine which Digital Receptionist will handle the call.

To keep the system simple, I used only two outgoing messages. The first, titled Night, says: "We are unavailable at this time, please press 1 to leave a message, or if this is an urgent matter press 0 to ring all the phones." The second message, titled Weekday, says: "Thank you for calling. To ring the house, enter 21; to ring the office, enter 22; to ring all the phones, enter 20 or stay on the line." Using these scripts, I select System

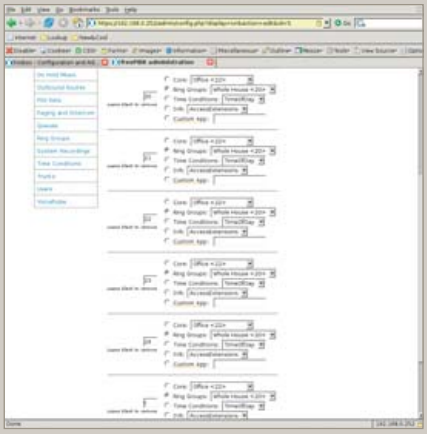


Figure 22. Finish setting up ring-all defaults.

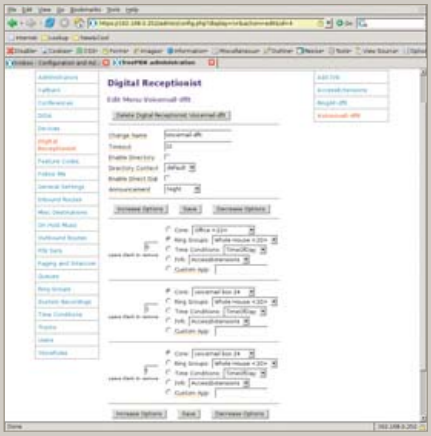


Figure 23. Set up voice-mail default.

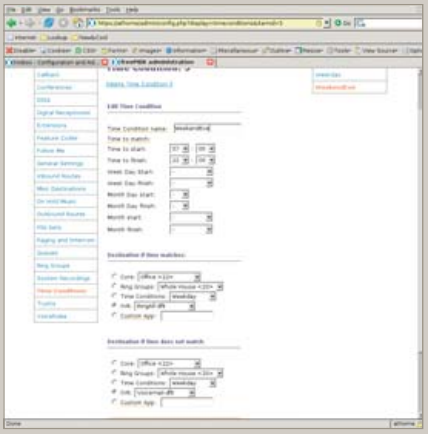


Figure 24. Set up time conditions on how the Digital Receptionist works.

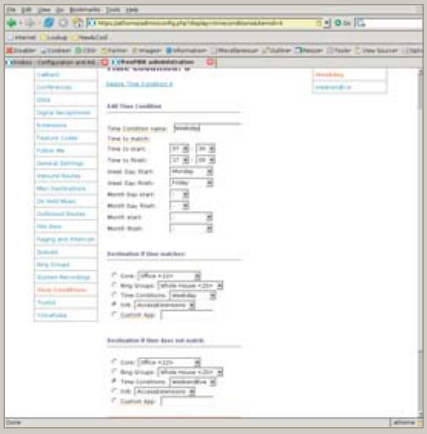


Figure 25. Time Condition for Weekdays

Figure 26. Inbound Route Configuration

Figure 27. Fax Handling for Inbound

Recordings from the left-side menu and select Add Recording to get to the first screen (Figure 18). It is possible to upload a WAV file directly, but I used an extension to record directly into the system. After clicking Go, I saw the screen shown in Figure 19 with instructions for recording the message. After recording each message to my satisfaction, I gave the recording its name and clicked Save.

The Digital Receptionist is only a bit more complex with three menus. I defined each of these by selecting the Add IVR link at the top of the right-side menu. The configuration of a Digital Receptionist menu is quite straightforward, once a person knows what each option does. The general section at the top allows for defining or changing the name of the IVR; next, is the number of seconds the caller has to enter an option, after which the t option is used; Enable Directory and Directory Context allow the caller to go to the automated directory system by entering #; Enable Direct Dial means that the caller can enter directly any extension defined in Extensions; Announcement is the audio message played to the caller before making the dial options available. Below the general section are other choices callers can select to take them to other parts of the system. Options for the caller to enter must not conflict with any Extensions. What can be chosen for destinations depends on the modules installed in the system, and most are self-explanatory. One of the possible destinations is another IVR menu that allows for very powerful cascading menu systems.

With this understanding of Digital Receptionists, we can look at the contexts I defined. Access Extensions is intended for the work/school day (Figure 20). It plays the Weekday announcement, allows direct dialing of any Extension, defines options to ring the Ring Groups, and if the caller does nothing (the t extension), all phones ring. RingAll-dflt is for evenings and weekends (Figures 21 and 22). It plays the same message, but it does not allow direct dialing any of the extensions. Rather, all extension numbers entered are redirected to ring all phones. The last IVR, Voicemail-dflt, is for calls arriving after all the children are in bed (Figure 23). The Night message is played to the caller, who can enter 0 to ring the whole house or enter 1 (or do nothing) to go to the family voice-mail box. Direct dialing extensions is not allowed. With those three message menus, the system was ready to handle all incoming calls.

Next, to define which of the Digital Receptionist contexts handles calls based on the time they are received, I used Time Conditions. There are three categories of time I want to differentiate: the normal workday, non-workday waking hours and everything else. I first defined the weekend and evening Time Condition (Figure 24) so that calls between 7 am and 10 pm (22:00) go to the RingAll-dflt IVR, and outside of

that time, calls are handled by Voicemail-dflt. Then, I created a Time Condition to handle the weekday times (Figure 25), which checks to see whether the time is during the work/school day, and if so, it passes the call to the AccessExtensions IVR. If it does not match, it is passed to the WeekendEve condition for further testing. So, if a call comes in on Monday through Friday and between 7:30 am and 5 pm (17:00) the AccessExtensions IVR handles it. If a call is not in that time frame, the WeekendEve Time Condition takes control. If the call is between 7 am and 10 pm any day of the week, the RingAll-dflt IVR handles the call; otherwise, the Voicemail-dflt IVR takes control.

The remaining segment for handling incoming calls is to decide what to do with calls when they arrive in the system. Inbound Routes examines the Dial-In Direct number and caller ID and direct the call accordingly. I have only one route (Figures 26 and 27) for all incoming calls, so I left the DID Number and CID fields blank, and the Destination is the Time Condition Weekday, which routes the call to the initial time condition.

The Inbound Route screen also allows for fax handling and setting a distinctive ring on SIP phones (but not for ZAP channels). For added security against phone solicitors, the Privacy Manager can be activated, requiring callers with no caller ID to enter their phone number before proceeding through the system. I do not have caller-ID service, so I left that off. I have found that the phone system itself deters many of the automated phone solicitations we used to get.

Handling the VoIP Trunk

The last thing to deal with is call routing to and from the VoIP trunk. To route calls from internal extensions to the trunk and calls from the trunk to my office extension, I needed to modify the dial plan files a bit, which can be done through the WebUI or from the command line. From the initial WebUI screen, I selected System Administration, Config Edit, and then extensions_custom.conf to bring up an editing window for that file. Adding the lines shown in Listing 1 to the from-internal-custom context allow all internal extensions to connect to my work VoIP trunk. Incoming calls from that trunk do not need to go through all the processing that calls from the PSTN line do, because they were already filtered by the office PBX. So, I also created the context custom-from-work (Listing 2), which does nothing but ring my extension (ZAP/2) whenever a call arrives. I attached that context to the trunk at the beginning of this article when I changed the context option for the trunk. So, now all calls coming from the office will ring my phone, and dialing #8 from any of our phones connects to the office VoIP system.

Conclusion

I now have a full-time answering system that meets my goals. I am able to call from one part of the house to another simply by dialing an extension or ring group. Because the system picks up the line before we hear a phone ring, we have cut our spam calls to almost none. If people call when it is late, they are reminded of the time and directed to voice mail, but urgent calls still can get through. I am able to make and receive calls from the office PBX as though I were in the same building as the system. All this and the only investment I had to make was the cost of the Digium TDM card and my time. I had the computer here already, and our house was wired for analog phones when we got here.

So, I found that Trixbox does not compromise much flexibility in gaining tremendous ease of use. There are still some features I would like on the system, however. Most notably, I would like the phones to ring differently for internal vs. incoming calls. I also would like my office phone to ring differently to differentiate my office calls from home calls, and my intra-office calls from my incoming office calls. Watch for those topics in a future article.■

Listing 1. How to Call a Work Trunk

```
; Call the work trunk
exten => #8,1,NoOp(Calling Work trunk) ; comment in log file
exten => #8,n,Dial(IAX2/Work/bat) ; dial the extension "bat" on the trunk
exten => #8,n,NoOp(After dial) ; another comment for the log file
exten => #8,n,Hangup() ; if we get to here, hangup the line
```

Listing 2. How to Direct Work Trunk to ZAP/2

```
; custom-from-work receives the calls from the work
trunk and directs them to ZAP/2
; we don't worry about VM because the work system
will get it [custom-from-work]
exten => s,1,NoOp(cxt: ${CONTEXT} - x: ${EXTEN} -
prio: ${PRIORITY} - cid#: ${CALLERIDNUM}) ; info
exten => s,2,DIAL(ZAP/2) ; ring indefinitely
exten => s,n,NoOp(cxt: ${CONTEXT} - x: ${EXTEN} -
prio: ${PRIORITY} - after DIAL)
exten => s,n,Hangup()
```

Michael George lives in rural Michigan (Pewamo, to be exact) with his wife and five children. He is a Systems Analyst for Community Mental Health in Lansing and also does OSS consulting and phone system deployment along with Ideal Solutions (www.idealso.com). He can be reached at george@mutualdata.com.

Resources

Nerd Vittles: **www.nerdvittles.com**

voip-info.org: **voip-info.org**

Asterisk@Home Handbook Wiki: **www.voip-info.org/wiki/view/Asterisk%40home+Handbook+Wiki**


intel
Leap ahead™
Remember when
the sky was the limit?
With Intel® Software Development Products, the Swinburne Center for Astrophysics is showing today's kids a universe filled with unlimited possibilities.
intel
Software
LINUX* APPLICATIONS HELP UNRAVEL THE ORIGINS OF THE UNIVERSE.
Swinburne University's Center for Astrophysics and Supercomputing in Melbourne, Australia, is helping develop the next generation radio telescope in order to collect enough data to perform modeling and simulations of our entire galaxy. Their goal is to make realistic, 3D, virtual-reality animations available to the general public, particularly school children. To do this, they used the Intel® compilers to deliver application performance improvements — saving them valuable development time and money[1]. Whether you build applications for physics or financial analysis, Intel® Software Development Products help your Linux* applications reach for the stars.
Download a trial version today.
www.intel.com/software/products/nolimits
Copyright © Intel Corporation, 2006. All rights reserved. Intel, the Intel logo, are trademarks of Intel Corporation or its subsidiaries in the United States and other countries. *Other names and brands may be claimed as the property of others. [1] Performance and benchmark information were provided by Swinburne University. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing.

HOW TO CONFIGURE SIP and NAT

Can you hear me now? Making VoIP work through a NAT gateway.

SEAN WALBERG

For all the technology behind Voice over IP (VoIP), you'd expect that it would work on every network, but this unfortunately isn't the case. Network Address Translation (NAT) is a common practice used in networks, and it doesn't play well with VoIP. Solving this problem requires an understanding of NAT, VoIP and your VoIP setup. This article focuses on the SIP protocol for VoIP and the Asterisk VoIP software, but the problems and solutions are applicable to most other situations.

NAT is used to hide multiple hosts behind a different set of IP addresses. As a packet leaves the gateway, the source address is rewritten to the new external address. When the return packet arrives, the gateway replaces the destination address with that of the originating host before sending the packet along its way. The gateway also can use the same external address for multiple internal hosts. The source port of the connection may be changed to ensure that the connection can be identified uniquely by the gateway, so that the return packets can have the proper address replaced.

This last scenario is also called Port Address Translation (PAT), or IP masquerading. It is what Linux and other home firewalls use to hide multiple internal hosts behind the single public IP address assigned by the carrier. The hosts must be hidden because they are using special, private, addresses that can't be routed on the Internet (such as 192.168.1.1). NAT solves the connectivity problem by giving the host a proper source address on all external connections, which allows the remote host to respond. The downside is that the inside must originate all connections in order to build the translation table entries required for NAT to work.

The problem with VoIP and NAT is that both ends of the conversation have to be able to initiate a connection to each other. Consider the simplified sequence of events that happens when PhoneA calls PhoneB using their respective SIP servers, PBXA and PBXB.

1. PBXA sends a SIP invitation to PBXB on PhoneA's behalf. In this invitation, it is PhoneA's IP address.

2. PBXB invites PhoneB to the conversation specifying PhoneA's IP address as the other end.

3. If PhoneB accepts the call, PBXB responds to PBXA with an acknowledgment that includes PhoneB's IP address.

4. PBXA tells PhoneA about PhoneB.

5. PhoneA sends audio using the Real-Time Protocol (RTP) to PhoneB.

6. PhoneB sends audio using RTP to PhoneA.

NAT can cause problems in several places. If one of the PBXes is behind a NAT gateway, the other PBX won't be able to contact it without some additional network setup. If one or more of the phones are behind a NAT gateway, the other phone will be trying to send audio to a non-routable address. This results in failed calls or missing audio.

Asterisk calls the handing off of the phone call in steps 2 and 4 above a re-invite or a native bridge. The steps above show that the phone talks to its local PBX, which in turn talks to the remote PBX. The local PBX then re-points the two sides of the call to each other, so that the local phone is talking to the remote end. Ideally, both sides will do this, and the phones are free to talk directly, leaving the SIP server out of the conversation.

The alternative to a re-invite is to have the PBX relay the voice packets between the two endpoints. We look at this in more detail later, but first, here's a more common scenario.

The simplest situation is when a SIP client is behind a NAT gateway connecting to a server on the Internet. The client creates the translation entry for the SIP traffic when it first registers. As long as there is frequent communication between the two hosts, such as one packet per minute, the channel will stay open. The only configuration needed is to have the client use its external address in all SDP packets. On clients that support it, enable STUN (Simple Traversal of UDP through NAT), so the client can determine the external address dynamically, or enter it manually. Asterisk doesn't support STUN at this time, so all NAT configuration must be done manually. The following commands in /etc/asterisk/sip.conf set up the NAT properly:

```
[general]
localnet=192.168.0.0/255.255.0.0 ; or your subnet
externip=x.x.x.x                 ; use your address

[YOURREMOTEPEER]                 ; your peer's name
nat=yes
qualify=yes                      ; Force keepalives
```

With this configuration, Asterisk uses the address defined by externip for all calls to the peers configured with nat=yes. The addition of qualify=yes causes Asterisk to test the connection frequently so that the nat translations aren't removed from the firewall. With these two commands, there always will be a communications channel between Asterisk and the peer, and Asterisk will use the outside address when sending SDP messages.

If you have multiple phones and an Asterisk server behind a NAT gateway, the solution gets more complex. Calls between the phones will work fine because NAT isn't needed. For calls between you and other systems on the Internet though, there will be problems. Unless you register to the remote side as a client (as done in the previous example), you will not be able to receive SIP messages, so you will not be able to accept calls. Second, the address information in the call

setup will point to the internal address of the phone, and the one-way audio problems mentioned previously will crop up.

The easiest solution to this is to avoid NAT entirely. This may seem out of place in an article dealing with NAT, but if you have a public IP address available for your call server, use it! If your Asterisk server is connected to both the Internet and the internal network, the SIP port is reachable from both the inside and the outside, and the only problem is ensuring RTP flows properly. The PBX server need not be configured to route between the interfaces or provide masquerading; it simply needs to bridge the inbound and outbound voice calls.

As I mentioned earlier, the PBX either can stay in the voice path or get out of the way. In the latter case, the PBX tells both endpoints about each other after which the endpoints talk directly. However, Asterisk could have a call setup with both endpoints and relay the RTP packets on behalf of each endpoint. The inside host would be talking to the inside address, and the outside host would be talking to the outside address. The only configuration required to achieve this in sip.conf is to disable re-invites:

```
[general]
canreinvite=no      ; force relaying
```

This configuration works well because the Asterisk server can speak freely to the Internet to send and receive calls. It also can talk to the internal phones, and by some simple bridging, completely ignore NAT.

As it turns out, this relaying behavior also is required when the Asterisk server has only a private address. The RTP ports will have to be forwarded on the firewall too. RTP chooses random port numbers based on configured limits. Before the ports can be configured, they should be limited in range. Configuring the firewall rules is much easier if the range of ports is known beforehand.

The range of ports to be used for RTP is defined in rtp.conf. The following configuration will limit Asterisk's choice of RTP ports from 10000 to 10100:

```
[general]
rtpstart=10000 ; first port to use
rtpend=10100   ; last port to use
               ; rounded up if odd
```

Asterisk will need several RTP ports to operate properly. Only even ports are actually used, and disabling of re-invites causes two connections to be built per call. These ports and the SIP port must then be forwarded in by the firewall. The iptables syntax is:

```
iptables -t nat -A PREROUTING -i eth0 -p udp \
-m udp --dport 10000:10100 -j DNAT \
--to-destination 192.168.1.10
iptables -t nat -A PREROUTING -i eth0 -p udp \
-m udp --dport 5060 -j DNAT \
--to-destination 192.168.1.10
```

Replace eth0 with the outside interface of your firewall and 192.168.1.10 with the address of your Asterisk server. These rules tell the Linux kernel to translate the destination address of any UDP packets in the given range that are entering the outside interface. This must happen at the PREROUTING stage as opposed to the POSTROUTING stage, because the destination address is being translated. At this point, any SIP or RTP packet from the Internet will be forwarded to the internal Asterisk server for processing.

When a remote station makes a call to Asterisk, the SIP packet will be forwarded in because of the iptables rules. Asterisk will stay in the media stream because of the canreinvite=no command, and it will use the external address of the firewall in any SDP packets because of the NAT commands. Finally, the media stream will be forwarded to the Asterisk server because of the combination of iptables RTP forwarding and port ranges defined in rtp.conf.

Up to this point, the configuration has focused on getting Asterisk working behind a NAT gateway, with some extra details to make the phones relay through Asterisk. There are, of course, more general solutions.

As I said earlier, if you can avoid NAT in the first place, it's in your best interests to do so because it avoids all the problems encountered so far. The Asterisk gateway can have a very restrictive firewall policy applied to it—all that is needed is to allow UDP 5060 for SIP and whatever port range is defined in rtp.conf. In this configuration, Asterisk can contact both the internal phones and the rest of the Internet.

The most promising solution to the NAT problem is to have the firewall rewrite the SIP body as it translates the source address. The address specified for the RTP session would be replaced by the firewall itself, which also would take care of forwarding the RTP stream once it arrives. Some commercial firewalls do this. Linux iptables have shipped with ip_nat_sip and ip_conntrack_sip modules since kernel version 2.6.18. These modules are designed to take care of translating SIP, but after extensive testing, I was unable to get it working completely. I had success with inbound calls from a VoIP provider with re-invites disabled, but that was it.

IP or GRE tunnels (unencrypted) and IPSec VPNs (encrypted) are another option for getting around the need for NAT. Multiple sites are connected with a tunnel that encapsulates the internal traffic in another IP packet using the gateway's address as it leaves the outside gateway. The encapsulation is removed at the destination end. This is helpful only if you set up the tunnels beforehand. Because VPNs also are used to connect branch office data networks, this option already might be available to you. The issues of fragmentation that plague data applications aren't a problem for VoIP because small packets are used.

If SIP is not a requirement, and you're using Asterisk, consider using the IAX protocol. IAX tunnels both the control traffic and the voice traffic over a single UDP conversation that can be port-forwarded, filtered or translated easily. This method is limited to a static set of tunnels, which is sufficient if you're connecting some PBXes over the Internet or connecting to a long-distance provider.

Sometimes the above solutions aren't available to you. In that case, it might be advisable to move to a full-featured SIP proxy and use Asterisk only for voice applications, such as voice mail. SIP Express Router (SER) is a powerful SIP server that handles NAT well and is used by several high-volume services, including Free World Dialup. SER's job is only in setting up calls between endpoints, so it must rely on other applications, such as specialized media proxies, to handle RTP streams if needed.

The step beyond a SIP proxy is a Session Border Controller (SBC), which is like a VoIP firewall. The SBC can intercede in either the signaling or RTP paths to add extra features, such as signaling protocol or codec translation, all while enforcing security policies. These are almost exclusively commercial products.

It is no secret that problems will be encountered when rolling out VoIP, especially when the Internet and NAT are involved. Understanding how the protocols involved work is the first step to solving these problems. You've now seen some of the solutions, from reconfiguring the phone or PBX to port translation to additional products or even an adjustment to the architecture. With these problems out of the way, you are free to enjoy the benefits of VoIP.■

Sean Walberg is a network Engineer from Winnipeg, Canada, and has been working with VoIP for several years. You can visit him at ertw.com.

Hear Yourself Think Again!

WhisperStation™

Cool... Fast... Silent!

For 64-bit HPC, Gaming and Graphic Design Applications

Originally designed for a group of power hungry, demanding engineers in the automotive industry, WhisperStation™ incorporates two dual core AMD Opteron™ or Intel® EM64T™ processors, ultra-quiet fans and power supplies, plus internal sound-proofing that produce a powerful, but silent, computational platform. The WhisperStation™ comes standard with 2 GB high speed memory, an NVIDIA e-GeForce or Quadro PCI Express graphics adapter, and 20" LCD display. It can be configured to your exact hardware specification with any Linux distribution. RAID is also available. WhisperStation™ will also make a system administrator very happy, when used as a master node for a Microway cluster! Visit www.microway.com for more technical information.

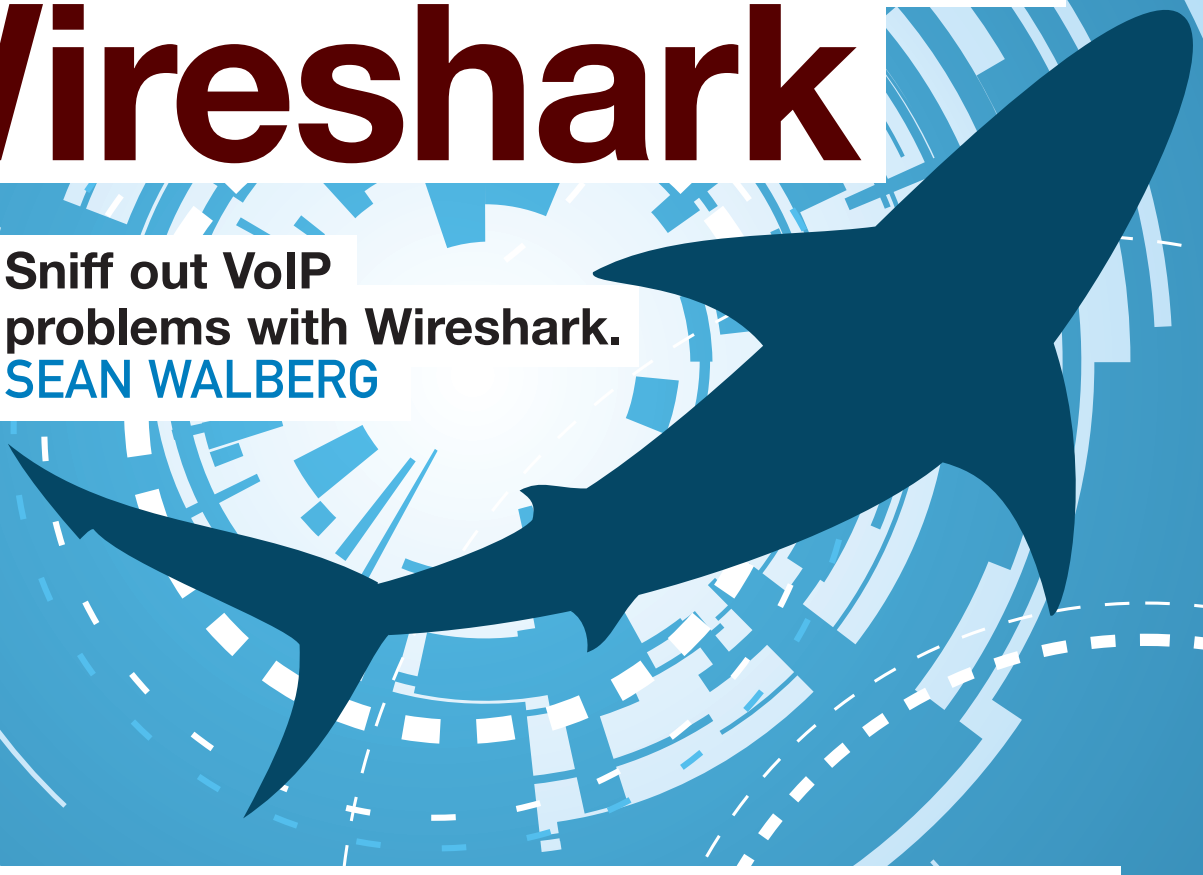
Experience the "Sound of Silence".
Call our technical sales team at 508-746-7341 and design your personalized WhisperStation™ today.

AMD 64 Opteron

intel inside XEON

Microway®
Technology you can count on™

EXPOSE VOIP PROBLEMS USING Wireshark



Sniff out VoIP problems with Wireshark.

SEAN WALBERG

VENDORS WOULD HAVE YOU BELIEVE you need to spend thousands of dollars on protocol analyzer software to troubleshoot Voice over IP (VoIP) networks. After all, voice quality suffers if packets are late, missing or out of order. Fortunately, Wireshark is an open-source package that can do everything these tools can and more.

Wireshark, formerly known as Ethereal, is a network protocol analyzer. Its job is to listen to network traffic, display it in a format that makes sense and then help you find problems. VoIP involves a complex set of protocols that Wireshark can decode and relate to each other. For example, the procedure to set up a call involves a different protocol than the voice traffic itself. Wireshark uses the information from the call setup to better understand the voice flow. With this data in your hands, you can isolate the cause of VoIP problems.

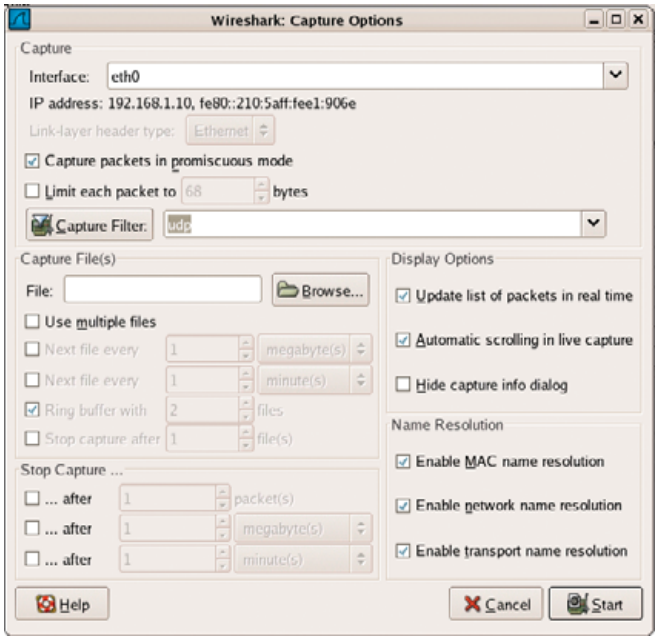


Figure 1. Wireshark Capture Options Dialog Configured for VoIP Analysis

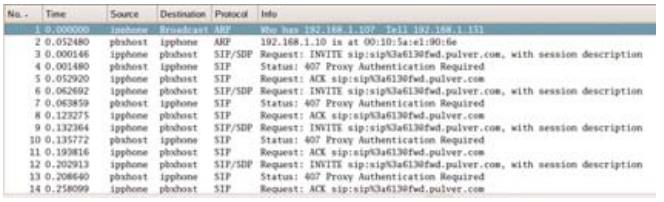


Figure 2. Wireshark Capture of an IP Phone's Traffic

To find the cause of VoIP problems, you must be able to follow the flow of calls from start to finish and ensure that the correct events are happening. This requires an understanding of both the underlying protocols and some telephony concepts. Fortunately, Wireshark provides some excellent tools to help interpret the data. The rest of this article focuses on using Wireshark's tools to solve three common VoIP problems.

The first example is of a phone that's not working—every time a number is dialed, the phone idles, and no ringing is heard in the earpiece. Wireshark is used here to look at the traffic between the phone and the PBX.

Launch Wireshark as root, and select Capture→Options to bring up the Capture Options dialog, as shown in Figure 1. Ensure that the correct interface is being used; otherwise, you won't see the traffic. In this example, I am capturing traffic on the PBX's only interface, eth0.

Before you capture, you have the option of specifying a capture filter to limit the number of packets you have to deal with. Because both the signaling traffic (SIP) and voice traffic (RTP) are UDP-based, I specify udp as a capture filter. As a convenience, I also check the Update list of packets in real time option so I can verify I'm getting the proper traffic. Click the Start button to start capturing, and soon the upper pane will show several packets (Figure 2).

Figure 2 shows the packet list. The first four columns show the packet number, time and the addresses of the hosts involved. Because Wireshark understands SIP, it is able to identify the packets as SIP in the Protocol column and give detailed information in the Info column.

A cursory examination of the information in Figure 2 shows that the same three packets repeat themselves four times. First, the ipphone host sends a SIP INVITE message to pbxhost, which is used to open a conversation. Next, the PBX responds with "407 Proxy Authentication Required" to which the phone responds with an ACK, and then tries again in packet 6. Because this process repeats itself continuously, it's a good guess that the phone is trying to authenticate with the wrong user name and password.

The first example made use of Wireshark's high-level interpretation of individual packets to help highlight a simple problem. The next example requires a more holistic view of a VoIP call. Here, Wireshark is used to view the SIP call setup and then identify the critical packets. In this example, a phone is experiencing one-way audio. Outbound phone calls ring and are picked up, but only one party hears audio. The conversation was captured in the same manner as in the previous example, resulting in 140 frames. This is too many to look at without some help, so some of Wireshark's advanced analysis tools are used.

With the capture loaded, select Statistics→VoIP calls. Wireshark analyzes the capture for any VoIP-related packets and provides a summary on the screen. Click on the call (if there are multiple calls shown, hold down the Ctrl key to select additional calls), and click on the Graph button. You will see a summary of the SIP call, as shown in Figure 3.

The Graph Analysis dialog shows the SIP messages sent by the various parties. The first message in Figure 3 is an INVITE message, which is the first step in setting up a call. In this case, the IP phone is asking the PBX to place a call to a SIP address of sip:613@fwd.pulver.com, which is an echo service provided by Free World Dialup for testing SIP calls. The response to this is a request for authentication, which is

Monitoring: It's All About Location

If some of the packets you are looking for don't reach the network interface of the machine running Wireshark, you've got a problem. Switched networks, by design, try to limit the traffic only to the ports involved with the communication in order to improve performance. Spanning trees and IP routing only complicate the matter of trying to find out where to put your probe.

Your best bet is to capture traffic directly on one of the machines involved. If you can't get Wireshark on that machine (it runs on Windows too), you always can capture traffic locally with tcpdump and ship the trace to another machine for analysis. If that isn't possible, or if the traffic spans multiple machines, it's time to get help from your network hardware.

Most managed switches offer a feature that will mirror all the frames from one interface to another. Thus, you could plug your Wireshark station in to a switch port with your Internet connection, mirror all the traffic from the Internet router to your workstation, and you'd be able to analyze all Internet traffic. Cisco calls this Switchport Analyzer or SPAN, HP calls it a Monitor Port, and 3Com calls it a Roving Analysis Port.

The choice of which port to mirror depends on your network, but generally, ports that connect to other switches or routers are a good bet. This is also where capture filters help reduce the background noise you have to deal with.

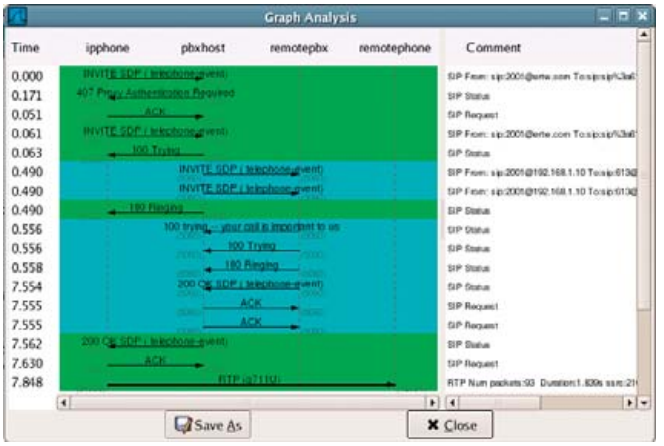


Figure 3. Wireshark's Graphical Interpretation of a SIP Conversation

```
▷ Request-Line: INVITE sip:6130fwd.pulver.com SIP/2.0
▷ Message Header
▽ Message body
  ▽ Session Description Protocol
      Session Description Protocol Version (v): 0
    ▷ Owner/Creator, Session Id (o): root 20990 20990 IN IP4 192.168.1.151
      Session Name (s): session
    ▽ Connection Information (c): IN IP4 192.168.1.151
        Connection Network Type: IN
        Connection Address Type: IP4
        Connection Address: 192.168.1.151
    ▷ Time Description, active time (t): 0 0
    ▽ Media Description, name and address (m): audio 34008 RTP/AVP 0 3 8 127
        Media Type: audio
        Media Port: 34008
        Media Proto: RTP/AVP
        Media Format: ITU-T G.711 PCMU
        Media Format: GSM 06.10
        Media Format: ITU-T G.711 PCMA
        Media Format: 127
    ▷ Media Attribute (a): rtpmap:0 PCMU/8000
    ▷ Media Attribute (a): ptime:20
    ▷ Media Attribute (a): rtpmap:3 GSM/8000
    ▷ Media Attribute (a): ptime:20
    ▷ Media Attribute (a): rtpmap:8 PCMA/8000
    ▷ Media Attribute (a): ptime:20
    ▷ Media Attribute (a): rtpmap:127 telephone-event/8000
    ▷ Media Attribute (a): fmtp:127 0-16
    ▷ Media Attribute (a): silenceSupp:off - - - -
      Media Attribute (a): sendrecv
```

Figure 4. Wireshark's Display of a SIP Packet Containing an SDP Message

acknowledged. The phone tries again and is given a status of "Trying" by the PBX. The PBX then proceeds to INVITE the remote endpoint by contacting fwd.pulver.com. Several more messages are exchanged before the call is set up properly.

The problem at hand is one-way audio. Indeed, the Graph Analysis window shows that the IP phone sent Real-Time Protocol (RTP) voice data to a SIP endpoint on the Internet, but it does not show a stream in the reverse direction. To determine why no audio was sent back, it is necessary to read deeper into the SIP packets.

One of SIP's jobs is to set up the RTP stream between two endpoints. It does this through the Session Description Protocol (SDP), which carries the information about codecs, IP addresses and port numbers that is necessary for VoIP to work. Note that the endpoints speaking SDP need not be the ones talking to each other! In Figure 3, pbxhost and fwd.pulver.com are negotiating with each other over the Internet, but they each specify a different endpoint to terminate the voice call. The RTP streams are unidirectional, so a full duplex conversation requires a separate RTP stream to be set up in each direction, using two separate SDP messages.

With that in mind, it seems prudent to find out the SDP information sent from pbxhost to fwd.pulver.com. This SDP message will contain the IP address, UDP port and codec that the remote end is to use to talk to the local IP phone. SDP messages are tagged with SDP in the Graph Analysis window to help you spot them. This packet is at offset 0.490 in Figure 3. When you click on the message in the Graph Analysis window, the corresponding packet is highlighted in the packet list pane of the main Wireshark window.

Wireshark also decodes the protocols contained within the currently selected packet in the packet detail pane, in addition to providing a summary of the packets in the packet list pane. Decodes are separated for each layer, such as Ethernet, IP, UDP and SIP. You can dig into the fields by clicking the arrows on the left side of the window. Figure 4 shows the SDP message from above with some of the relevant fields expanded.

Each line in an SDP message describes a particular attribute of the session to be created and follows a simple attribute=value format, where the attribute is a single letter and the value is a text string. Wireshark uses its understanding of the protocol to add some extra text, such as descriptions of the attributes. For VoIP sessions, the important attributes are:

- a—attributes, such as codecs and silence suppression.
- c—connection information, including the IP address expecting the RTP stream.
- m—media description, including the port number on which the RTP endpoint will be listening.

From the Connection Information (c) line, you can determine that the host being sent is 192.168.1.151. The m attribute specifies that the RTP stream is expected to be on port 34008. Looking through the attributes, the PCM u-law and a-law codecs are offered, along with the compressed GSM format and touch tones via the telephone-event media format. The problem with this offering is the IP address 192.168.1.151 is a private address, unreachable from the Internet. When the remote host tries to send packets to 192.168.1.151, the packets are lost because no such route exists on the Internet.

For the sake of completeness, the SDP information for the other side of the conversation is at time 7.554. After the offer is acknowledged, the two phones begin sending voice packets to the address and port specified in the earlier SDP messages. Because pbxhost offered an invalid address to fwd.pulver.com, the other half of the voice call is never seen.

This illustrates a common problem when using VoIP through a NAT gateway. The problem can be solved several ways depending on one's needs, but that's another article!

So far, the troubleshooting has focused on the call signaling aspect of VoIP. Once the endpoints begin sending RTP streams, the effects of network quality can be heard in the audio.

Network conditions that affect VoIP are latency, jitter and packet loss. Latency is the time it takes for a packet to travel from point A to point B. Jitter is the variation in latency over a series of packets. Loss is the number of packets sent from point A that never make it to point B. Because VoIP audio is sent over real-time UDP, a packet that arrives out of order has to be discarded if the packet ahead of it already has been played.

For the final example, I made a call and captured the conversation including the call setup. Wireshark uses the SIP information to get more details about the RTP packet stream, which enables the RTP analysis tools to be used.

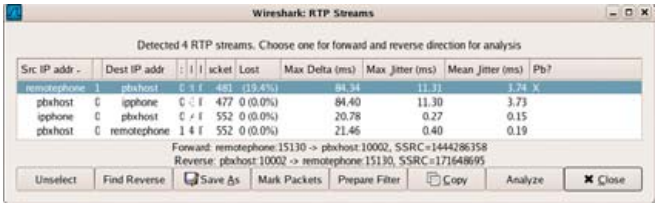


Figure 5. Wireshark's List of Found RTP Streams

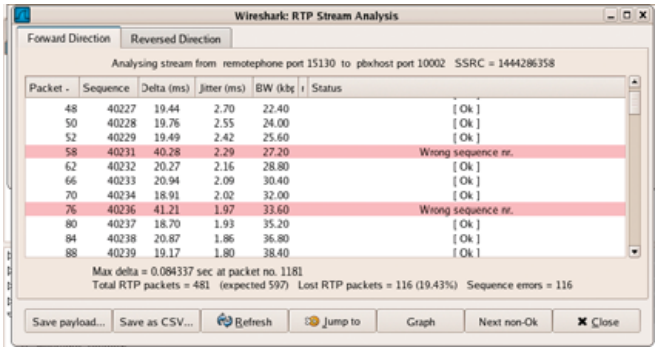


Figure 6. RTP Stream Analysis of a Stream with Excessive Packet Drops

Select Statistics→RTP→Show All Streams. Wireshark uses the decoded packets to provide a list of all the audio conversations and some basic statistics, as shown in Figure 5.

Figure 5 shows four streams, because each audio stream comprises two streams in opposite directions, and the PBX was bridging the connection between the two phones. The first indication of a problem is the final field, which shows an X when there are VoIP-related problems. The Max Delta (latency) was 84ms for that stream, which is good. The Max Jitter was also good at 11ms (150ms of one-way latency and 20ms of jitter are the limits of what's considered acceptable). However, nearly 20% packet loss was encountered, which is extraordinary!

Select the problematic stream for further analysis by clicking on it. After that, click on the Find Reverse button to select the other half of the conversation. Finally, click on Analyze to provide a packet-by-packet look at the stream. Lost packets will show up as having the wrong sequence number. This screen, shown in Figure 6, also displays helpful statistics, such as the current bandwidth, latency and jitter.

Figure 6 clearly shows some network problems, because the packet loss is not normal. (The steady bandwidth and regularly spaced packet drops are strong indicators of priority queue overruns or policing in network equipment.) This loss will not be seen in the reverse direction, because the capture was taken close to the source of the conversation. You can save the contents of the window with the Save CSV button.

Network problems are more difficult to solve, because they require interaction with network equipment and potentially with other parties. If your VoIP environment doesn't use the Internet, you can capture packets at various points on the network to find where the poor conditions are introduced. If the call flows over the Internet, you must investigate your connection to the Internet and possibly work with your carrier.

The RTP analysis can be used with proprietary systems with some extra configuration. If Wireshark doesn't understand the signaling, it won't be able to decode all of the RTP information. Select Edit→Preferences→Protocols→RTP, and check the Try to decode RTP outside of conversations box. Analysis will be a bit slower, but Wireshark will try to determine whether every UDP packet belongs to an RTP stream, allowing you to use the RTP tools to look at the call.

The final feature we examine here is the ability to listen to the contents of the voice call. With the Stream Analysis window still open, select the Save Payload button, select the .au file format, and provide a filename. After pressing the OK button, the voice call itself is saved to your hard drive. The resulting file can be played in XMMS, soxplay or some other audio program.

Troubleshooting VoIP is unlike most other network troubleshooting, because of the separate signaling and voice protocols, and the real-time nature of voice traffic. These three examples have shown the important features of Wireshark that deal specifically with analyzing the SIP call setup and RTP voice streams. These features rival those of commercial products and enable to you find the root of VoIP problems.■

Sean Walberg is a network Engineer from Winnipeg, Canada, and has been working with VoIP for several years. You can visit him at ertw.com.

Resources

Wireshark: **wireshark.org**

Telecom Terms and Concepts: **www.oreillynet.com/pub/a/etel/2006/02/07/telecom-terms-and-concepts.html**

FWD: **www.freeworlddialup.com**

Applying Adhearsion to Asterisk

Tackle your sticky VoIP projects with Ruby and Adhearsion.

JAY PHILLIPS

Following the trend of other corporate servers shifting from closed-source platforms like Windows NT and UNIX, VoIP too now booms with an exodus to the open-source Asterisk PBX and Linux. In its own picture, Asterisk looks readier than ever to meet these needs. The great folks at Digium, Asterisk's maintaining company, stress product quality and lead VoIP innovation.

Integrating technologies with VoIP also makes the picture even prettier. With standard computers crunching the calls, connecting the dots plainly makes sense. Traditional ways to integrate Asterisk really exist as hacks though, not as comprehensive solutions.

Enter Adhearsion

This new open-source framework takes on integration issues ranging from enterprise must-haves to living-room hacking projects. Written in Ruby, Adhearsion comes out of the box with more than what many companies spend thousands on. Though its name derives from being "adhesion you can hear", it can work even without Asterisk when trying to splice two or more technologies together outside of the VoIP picture frame. Because Ruby and Adhearsion both aim to improve productivity, VoIP systems now more than ever make fun projects for your free time or effective tools in bringing your employees, customers and the world closer.

When a call comes in on an Asterisk server managed by Adhearsion, Asterisk acts as a metaphorical kernel for the call by managing low-level hardware, converting codecs, canceling echo and so forth. For the logic it should perform during the call, such as playing messages, connecting through to someone or taking input, it establishes a TCP connection to a running Adhearsion dæmon and receives commands one at a time, executing each and sending back a response.

Let's get a working Adhearsion box running to demonstrate.

Up and Running

Adhearsion, like most other Ruby projects, uses the fantastic package manager RubyGems. If you read *Linux Journal*'s July '06 issue on Ruby or have picked at the language, you've likely bumped into this great software. If not, package managers for any popular Linux distro barely differ. The Adhearsion installation process constitutes installing Ruby and RubyGems, then pulling Adhearsion down with a `gem install` command.

The installation example below uses Ubuntu/Debian's apt-get package manager. You, therefore, may need to make minor modifications to the package names for your distro. If you don't have a RubyGems system package, drop by its Web site (see Resources) and pick up the tarball:

```
apt-get install ruby1.8 ruby1.8-dev irb1.8
tar zxf rubygems*.tgz
cd rubygems*
ruby setup.rb
gem install adhearsion --include-dependencies
```

With the last command, RubyGems created the ahn (pronounced Anne) command.

You can use ahn to create new Adhearsion projects, execute an Adhearsion process normally or as a dæmon, install open-source framework extensions remotely, read help documentation and so forth. Type `ahn help` for more information.

This guide assumes you have an existing Asterisk server configured. If you would like more information on how to do this, see the Resources section. It takes adding only a simple instruction to the extensions.conf configuration script to have Asterisk utilize Adhearsion. If you had your phone users routed to your "internal" context, you would modify the "internal" context's definition as such:

```
[internal]
exten => _X.,1,AGI(agi://192.168.1.2)
```

Next, create a new Adhearsion project with the command:

```
ahn create ~/helloworld
```

This generates an entire directory structure with which you can mingle safely. To start up Adhearsion, supply the path to your Adhearsion directory to ahn:

```
ahn start ~/helloworld
```

Read on to learn ways you can modify this new project to suit your interests.

Make Call Instructions Fun

Traditionally, engineers route Asterisk calls in one of two ways: modifying the extensions.conf file or writing an Asterisk Gateway Interface (think CGI) script. Most generally use the former but, in situations where integration matters, one may use them together. Typically, developers write AGI scripts on the filesystem of their Asterisk machine, which communicate plain text over STDIN and STDOUT.

But, trouble arises when this scales both in complexity of code and complexity of use. Because Asterisk executes this script as a subprocess, it actually builds up whatever overhead is needed for the AGI program (for example, the Perl interpreter or a database socket), executes the code and then tears it all down again. Although tolerable in smaller situations, processes carry greater tolls on memory and running time as concurrent use increases. Additionally, without a framework, scripts in this way also contain suboptimal amounts of repetition.

Adhearsion utilizes Asterisk's FastAGI feature for managing calls from another IP-addressable machine. In this way, precious RAM and CPU cycles receive virtually no impact on the Asterisk box and, just as important, Adhearsion has already established the overhead for each call before it comes in.

With your new example Adhearsion project created earlier, open your extensions.rb file. Below, I've included an example file from

which you can get a feel for how Adhearsion works to handle calls. Take a minute now to read it over:

```
# File extensions.rb
internal {
  case extension
    when 101...200
      employee = User.find_by_extension extension
      if employee.busy? then voicemail extension
      else
        dial employee, :for => 10.rings
        voicemail unless last_call_successful?
      end
    when 888
      play weather_report("Dallas Texas")
    when 999 then +joker_voicemail
  end
}

joker_voicemail {
  play %w"a-connect-charge-of 22
            cents-per-minute will-apply"
  sleep 2.seconds
  play 'just-kidding-not-upset'
  check_voicemail
}
```

If you feel adventurous, try using this now with any phones on your Asterisk box.

Note that this is all valid Ruby. Ruby fundamentally permits the modification of how aspects of the language function. As such, Ruby code can become modified for a particular need or particular domain of needs—hence the name commonly used in the Ruby scene: domain-specific language or DSL. The full benefit of the Ruby language remains, however, giving it the best of both worlds.

Because our example Asterisk extensions.conf from above invokes Adhearsion within a context named internal, Adhearsion maps this directly over by convention. Adhearsion creates many call-related variables as simple local variables before it processes a context. The case statement for the auto-generated extension variable used shows Ruby's great support for ranges of numbers, but a Perl-like regular expression literal could very well replace this.

The next line `employee = User.find_by_extension extension` may prove surprisingly subtle. You easily can infer there exists some collection of users with extensions and some method of finding them by their extension. If ActiveRecord is new to you, you probably didn't realize immediately that this actually abstracts database access. Before diving into a more complete explanation of how this works, let's finish grokking this file.

Implementations for determining whether the user should be disturbed can vary, but we need to know only a yes or no answer. Users are allowed to remain focused by having callers sent silently to their voice mail.

Users dialing 888 have a weather report converted from an Internet source to a series of played sound files that come with the standard asterisk-sounds package. If they dial 999, things get fancy—the plus sign before joker_voicemail does just what you might think: it executes the joker_voicemail block.

This joker_voicemail section uses `%w""`, a fantastic Ruby literal for an array of words. Automatically, Ruby will break apart this string by white-space, creating an array containing a series of sounds intuitively named for the speech they produce. But you ask, "What about this 22 here?"

When Adhearsion encounters numbers, instead of executing the Playback() application, it uses SayNumbers(). This breaks the number into the words twenty and two—two words for which sounds do exist. The end result contains no commas, no quotes between indices and no specification of the application used to play the file. Why should it?

You can find more detailed, working dial plans and documentation freely on the Adhearsion Web site. See the Resources section for more information.

Powerfully Use Your Database Software

If you've checked out Ruby on Rails, you likely know of its rock-star object relational mapper ActiveRecord. Instead of keeping ActiveRecord deeply intertwined with the Rails framework, the brilliant folks over at 37signals allow anyone to utilize this fantastic library through a gem. When you install Adhearsion, the `--include-dependencies` part pulls this in for you.

From a database modeling perspective, typical VoIP applications have users and groups of users. Therefore, in our database—whether you use MySQL, PostgreSQL, SQLite or nearly any other popular relational database management system—it makes sense to have a table called users and a table called groups. Because each record represents an individual user or group in the database, it also makes sense to have a User object and a Group object. Beginning to see how an object relational mapper applies here?

Here's a tangible, tasty example:

```
class User < ActiveRecord::Base
    belongs_to :group
    validates_uniqueness_of :extension
    def busy?
      # Implemented how you wish
    end
end

class Group < ActiveRecord::Base
   has_many :users
end
```

Ruby Methods

Notice in the example how Ruby methods generally differ from other programming languages. Calling a method does not require the parentheses when no ambiguity exists. This may seem strange at first, but if you can convince yourself to accept it, you may just find reading and writing Ruby code somewhat enjoyable.

Also, Ruby allows question and exclamation marks to conclude a method name to better express meaning. This follows Ruby's guiding Principle of Least Surprise to identify methods typically used to represent some condition on which the user may act or, with the exclamation mark, to signify optionally that the method is destructive to the variable.

This is where the magic happens. With so little code, ActiveRecord can actually make a lot of very logical conclusions. Given the class names User and Group, it finds their appropriate table by lowercasing them and then finding the plural forms users and groups, respectively. If we had used the class name Person instead of User, ActiveRecord would have assumed its associated table to be people.

To help ourselves out, we've told ActiveRecord that a User belongs to a Group and that Groups have many users. Given this, the foreign key to which a record in users maps is assumed to be group_id by default. With this identified, one can retrieve the Group object to which a User instance belongs simply by calling jay.group. If the jay variable belonged to the Group codemecca, jay and any other potential variables could be retrieved by codemecca.users.

So let's take a look at the dial plan example above. We're calling a method on User by the name of find_by_extension. Did we have to create that method anywhere here? No. But why not? ActiveRecord actually created the method, because it peeked inside the users table, found the names of its columns and created methods like find_by_extension. Explicated into a MySQL select statement, it would read `SELECT * FROM users WHERE EXTENSION='somenumber' LIMIT 1;`. Nice shortcut, eh?

Use VoIP and the Web Together to Collaborate

In a corporate environment, businesses depend on collaboration, and people look for any way to improve it with good reason. Adhearsion offers a free collaboration tool that, in the spirit of adhering everything together, takes traditional collaboration a step further.

Modern IP desk phones generally have a microbrowser that can pull down XML documents over HTTP and translate them to an interactive menu on the device. Although this may seem like a great technology, there's a significant caveat: every vendor's XML format differs, none document the feature very well and the available features vary vastly. Some vendors support images, HTML forms and a special URI for initializing a call, whereas others give you about three or four XML elements from which you can choose.

If vendors can't collaborate on consistency, something must abstract their differences (and quirks) by translating. Micromenus do exactly that with a few hints of additional cleverness.

Since the Micromenus sub-framework exists as a built-in helper, you manage the menus in the config/helpers/micromenus/ directory. When an HTTP request comes in over port 1337, Adhearsion takes the first argument in the URL's path as the desired filename. For example, if you programmed a phone to use http://192.168.1.228/main as its Micromenus URL, Adhearsion would attempt loading the file config/helpers/micromenus/main.rb. Modifying and creating files here takes effect even without restarting Adhearsion.

Using this example filename, let's fill it with some Micromenu code:

```
# config/helpers/micromenus/main.rb
call "Check your voicemail" do
    check_voicemail
end

call 505, "Call Support"

item "Join a conference" do
    call 'Join Marketing' do
        join :marketing
    end

    call 'Join Support' do
        join :support
    end

    call 'Join Ninjas' do
        join :ninjas
    end
end

item "Weather forecasts" do
    call "New York, New York" do
        play weather_report('New York, New York')
     end

    call "San Jose, California" do
    play weather_report('San Jose, California')
    end

    call "University of Texas at Dallas" do
      play weather_report(75080)
    end
end

item "System Administration" do
    item 'Uptime: ' + %x'uptime'
    item "View Current SIP users" do
        PBX.sip_users.each do |user|
            item "#{user[:username]@#{user[:ip]}"
        end
    end
end
```

This simple example, albeit very terse, produces an entire, respectable company intranet. Giving the item method a block automatically turns it into a link, abstracting the URL. Placing actual Ruby code in a submenu gives the menu behavior. Using the call method on a phone actually places the call to either the extension specified (without a block) or, because Adhearsion handles the call itself too, executes the dial plan behavior within the block when it finds the user generated a call through a Micromenu. This exemplifies the benefits of having both call routing and Web-based collaboration adhered this closely together in the same framework.

Create Framework Extensions

As any IP telephony engineer can affirm, taking someone else's VoIP functionality and merging it with your own is not for the faint of heart. Nor can you find VoIP functionality in abundance on-line. The sheer difficulty of reusing this kind of code severely discourages trading. Standard PBX dial plan configurations are typically meaningless without the half-dozen other configuration files that put them into perspective.

Not only does Adhearsion allow others easily to integrate with

their VoIP applications, it facilitates the integration. The Adhearsion Web site hosts a nice community where users can submit, tag, browse and rate extensions. All of these can be freely downloaded and copied to your helpers/ directory where Adhearsion will find it and automatically absorb its features into the framework. Extensions can vary from adding a new method to the dial plan DSL to entire Web servers that run in a separate thread.

Let's say in a fit of lunacy you decide to write a VoIP calculator application that speaks back an answer. Because you have many different mathematical operations from which to choose, you decide to implement a factorial method and expose it to the entire framework. This requires simply creating the file helpers/factorial.rb and adding the following code to it:

```
# helpers/factorial.rb
def your_factorial num
   (1..num).inject { |sum,n| n+sum }
end
```

When you start Adhearsion, you'll have the ability to use this in your dial plan, your Micromenus, Adhearsion's distributed computing servers, and any other nook or cranny of the framework. But, you say this doesn't cut it.

Like any dynamically typed language, this simply takes too long for very, very large numbers. Wouldn't it be nice if we could write this extension in C? Well, we can.

The terrific third-party library RubyInline takes a string of C/C++ code, read from a file or otherwise, and automatically compiles with the Ruby source headers, caches, and dynamically loads it into the Ruby interpreter. The library even finds any method signatures and creates a matching Ruby method. Static return types automatically convert to Ruby equivalents (int to Fixnum, double to Float) when the native code finishes its business. With this library, Adhearsion allows more efficient extensions to the framework with languages other than Ruby.

Because RubyInline requires the Ruby development headers and a configured compiling environment, it doesn't come in as an Adhearsion dependency. If you have GCC and its peripheral development requirements, do a `gem install RubyInline`, and throw this code in the file helpers/factorial.alien.c:

```
int fast_factorial(int num) {
     int sum = 0, counter = 1;
     while(counter <= num) {
          sum += counter++;
     }
     return sum;
}
```

Like other extensions, Adhearsion automatically finds this file and hooks its functionality into the framework. If the C code requires special compile instructions or include statements, you easily can add these to the helper's config file, which all helpers can optionally have. These config files exist in the config/helpers directory with the same name as the helper to which they belong but with the YAML .yml extension.

Let your imagination run away with you. If you come up with a great new idea for a VoIP system, the Adhearsion extension architecture serves as a great launchpad to materialize your concepts easily.■

Jay Phillips is a VoIP and Ruby enthusiast turned entrepreneur. As the creator and project manager of Adhearsion, Jay provides Adhearsion support and consulting through his newly created, Dallas-based company Codemecca, LLC.

Resources

Adhearsion Web Site: **adhearsion.com**

PwnYourPhone, Official Adhearsion Video Podcast: **PwnYourPhone.com**

Official Adhearsion Blog: **jicksta.com**

VoIP-Info Wiki: **voip-info.org**

Adhearsion Wiki: **docs.adhearsion.com**

Codemecca Web Site: **codemecca.com**

RubyGems Web Site: **rubyforge.org/projects/rubygems**

Combine uClinux and Asterisk on a Card

Embedding Asterisk on a Digikey Blackfin STAMP card.

DAVID ROWE

This article describes how to build an embedded Asterisk IP-PBX with four analog (FXO or FXS) ports. Total parts cost is about $500 US, which is competitive with PC/PCI-card based Asterisk solutions. Embedded solutions have the advantage of small size, low power and no moving parts. Figure 1 is a photo of the PBX hardware in action.

The PBX is built around a Blackfin STAMP development card, available off the shelf from Digi-Key for around $225 US. The Blackfin is a powerful DSP chip that runs uClinux. Sitting on top of the Blackfin STAMP card is a daughterboard, which contains interface hardware and the SD card socket. Into the daughterboard plugs FXO or FXS modules, one for each port. In this example, there are two FXS modules on the left and two FXO modules on the right. The color of each LED indicates the type of module inserted.

Here's a brief review of telephony jargon:

- FXO ports connect to telephone lines and the exchange.
- FXS ports connect to analog telephone handsets.

Figure 2 is a block diagram of the PBX hardware and software components. SIP phones or analog phones can be used for handsets (extensions), which are connected via a LAN to the IP PBX. External calls can be routed over the Internet or through the analog FXO ports. Internal calls between IP phones are routed over the LAN.

The PBX supports most of the features of Asterisk running on an x86 PC. As there is no hard disk, an SD card is used for voice-mail storage.

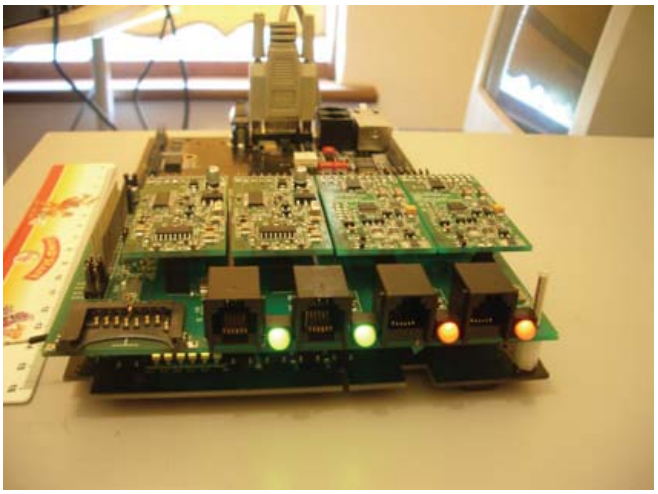


Figure 1. Digi-Key Blackfin STAMP Development Card

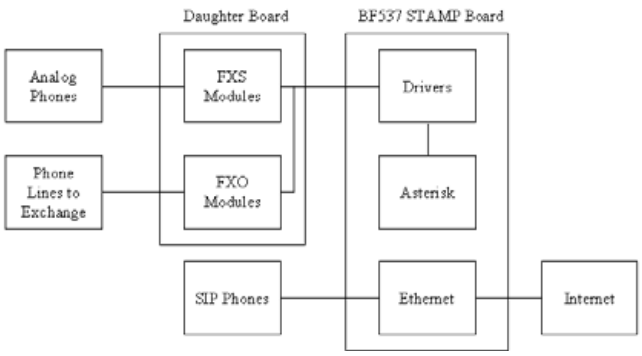


Figure 2. Sample PBX Configuration

Why Port Asterisk to uClinux/Blackfin?

I have a history of developing computer telephony hardware and have always wanted to build a small embedded box that combines a host processor, DSP, line interface hardware and software. It's an itch I've needed to scratch!

There are some very cool things about the Blackfin processor chip:

1. The problem with most embedded processors is that they are not very powerful. The Blackfin is a powerful host processor *and* a DSP—that is, it can run uClinux, Asterisk and codecs like G729 on the same processor at the same time. A standard 500MHz Blackfin runs at around 1,000 DSP-MIPs, which is plenty for codecs, echo cancellation and so forth.

2. A lot of effort and hardware cost is usually required to interface telephony hardware to the host processor (typically a PC), such as PCI bridge chips. The Blackfin makes it easy, as it has a lot of nice interfaces built in, such as serial ports, SPI and DMA controllers, which are all tightly integrated with the core processor.

3. The Blackfin chips are good value for the money, ranging from $4.95 US each (BF531 in 10k volume), which makes low-cost embedded telephony hardware a real possibility. This makes it possible to build an IP PBX including analog or E1/T1 line interfaces for far less than comparable PC-PCI card solutions.

4. Best of all there is an Open Source community that has developed GPL hardware (the family of STAMP boards).

Open-Source Hardware

The hardware designs for this project are open—the schematics and PCB layouts are freely available for anyone to download, copy and modify.

The hardware designs have been released under the GPL.

Although there is some debate over how defensible the GPL is when applied to hardware, the key ideas are similar to open-source software—the hardware designs are free as in speech, and a community exists that is working together on extending and enhancing the designs. Intellectual Property (IP) is shared for the mutual benefit of all.

The community is loosely organised under the Free Telephony Project, and it consists of private hackers, researchers and several companies who are donating time and other resources to the project. A series of hardware designs are being developed, for example, analog and ISDN interface hardware, and DSP motherboard designs. Significant software development work is also occurring, for example, open echo cancellation software and drivers for the hardware.

The outputs of the project are high-quality, professionally designed telephony hardware, freely available for all. Hardware development is a little different from software—hardware design/test cycles are much longer (for example, a bug might mean a new board needs to be manufactured), and, of course, it costs money to make a "copy" of a hardware design. However, the benefits of open hardware are similar to open software:

- Peer review is a wonderful way to trap bugs early, leading to big savings in development time.
- Re-use of open designs enables innovative products to be developed quickly and with a very small number of bugs.
- Discussion and contributions from people all over the world lead to a much higher-quality product than a product developed by one company in its own lab.
- Using open hardware, an individual or small company can build sophisticated telephony products without needing large company resources.
- The part I like best—you get to work with a community of talented hardware and software people! Some of the best and brightest minds out there seem to migrate naturally to open projects.

Business models are being developed that function within the open-hardware environment. For example, one company sponsoring our efforts has a service model based around low-cost telephony hardware—maintaining ownership of the hardware IP is not a critical part of its business plan.

Open hardware is also great for small, localised businesses and allows developing countries to build their own products locally—overcoming tariff barriers and building a local high-technology industry.

A *Really* Open Hardware/Software Project

This PBX runs open-source software (Asterisk), on an open-source operating system (uClinux). The operating system, drivers and applications are built using a GNU gcc toolchain.

The hardware designs are open and were designed using open-source tools. Schematic entry was performed using gschem, and the PCB layouts were developed using a program called PCB—both included in the gEDA package of open electronic design tools.

Verilog was used to implement some programmable logic on the daughterboard. Verilog is a language used to describe logic in electronic circuits. The Verilog code was developed with Icarus Verilog, also part of the gEDA package.

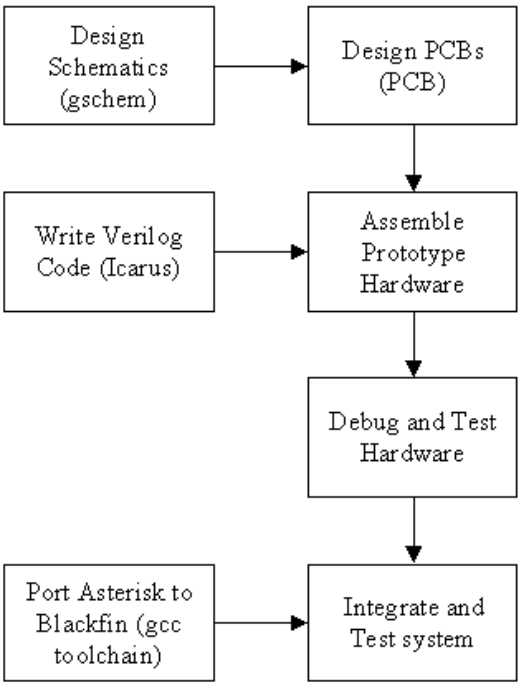


Figure 3. The Work Flow Used to Create the Project

Figure 3 shows the high-level work flow for the project. As you can see, hardware, software and even the tool used to design the hardware are all open.

Assembling the Hardware

Let's start putting it all together.

The first step is to plug your desired combination of modules in to the daughter card. For example, for a home PBX, you might simply want one FXO and two FXS modules; for an office you might want four FXO modules. In Figure 1, two FXO and two FXS modules are loaded.

The next step is to plug the daughter card/module assembly in to the Blackfin STAMP card. That's pretty much it for the hardware assembly. As a quick test, try applying power to the STAMP, and it should boot without any problems. Not much else will happen yet, as we need to compile and download the software.

Building the Software

The software is built on a Linux host PC, then downloaded to the Blackfin STAMP board using FTP (driven by some shell and Expect scripts). Here are the steps:

Download and install the Blackfin gcc toolchain. Detailed instructions for this step are on the **blackfin.uclinux.org** Web site.

You need a compiled version of the latest Blackfin STAMP uClinux distribution (uClinux-dist RPM or tarball). Before building it, set up the following configuration using `make menuconfig`:

- Kernel hacking: boot param—`root=/dev/mtdblock0 rw`. This makes the root filesystem read-write, which allows us to add the many files Asterisk requires to the root filesystem.
- Blackfin options: write back cache—this improves the speed of the DSP-intensive operations, such as the Speex codec, by about 10% per codec instance.
- Customize vendor/user settings: Flash tools—MTD Utils is switched off, as it breaks the uClinux-dist compilation (at least on my machine).
- I modified the file: uClinux-dist/vendors/AnalogDevices/BF537-STAMP/rc

to set the hostname and enable dhcpcd.

Generic instructions for building uClinux are included in the Blackfin uClinux-dist documentation. Read these and build uClinux. If your build was successful, a uClinux-dist/images directory will be created containing the uClinux images.

Download the uCasterisk tarball, then:

```
$ tar xvzf uCasterisk-0.1.6.tar.gz
$ cd uCasterisk-0.1.6
```

In the file .config, check that:

```
BR2_TOOLCHAIN_DIR="/opt/uClinux/bfin-uclinux/bin"
BR2_KERNEL_SOURCE="/opt/uClinux-dist/linux-2.6.x"
```

point to your toolchain and target kernel sources.

Then, typing:

```
$ make
```

downloads, patches and makes all the different packages you need. See the top-level Makefile for other useful options.

Downloading and Testing

Now, download Asterisk and all the support files to the Blackfin STAMP:

```
$ ./scripts/install_all stamp
```

where stamp is the hostname of your STAMP card. There will be a few pauses as the drivers are installed, and the LEDs on the daughterboard should light indicating that the FXS/FXO modules have been auto-detected.

The next step is to configure Asterisk for your combination of FXO and FXS modules. Asterisk configuration is a big subject and can be daunting for the beginner. To help get started, some basic configuration files have been developed.

Let's assume for now that you have loaded two FXS modules and two FXO modules. In that case, run:

```
$ ./scripts/config_2fxo2fxs stamp
```

Now, you are ready to start Asterisk. Use Telnet to open a console to the Blackfin STAMP:

```
$ /var/tmp/asterisk -vc
```

A lot of text will flash by as Asterisk boots, but eventually, you should be greeted by the Asterisk CLI prompt:

```
Asterisk Event Logger restarted
Asterisk Ready.
*CLI>
```

Expect and Embedded Systems

Expect is an extension to Tcl that lets you automate console sessions. I have found it to be very valuable for automating common steps in embedded systems work.

There are a couple of annoying problems with embedded system development:

1. On a regular desktop system, you compile code and then run it on the same machine. With an embedded system, you compile it on a host PC and then download to the host system—for example, using FTP or TFTP. This extra step becomes tedious and can lead to silly mistakes, like forgetting to download the latest version.

2. A lot of embedded development work involves device driver work, which can mean frequent crashes. During the development phase, embedded systems commonly use RAM-based filesystems, so a system crash or reset means you lose everything and must download all your software and config files all over again.

Expect lets you automate all this. For example, here is some Expect code to transfer a file via FTP to the target system:

```
#!/usr/local/bin/expect -f

set tarball [lindex $argv 0]
set target  [lindex $argv 1]

send "ftp $target\r"
expect "): "
send "root\r"
expect "Password:"
send "uClinux\r"
expect "ftp> "
send "cd /var/tmp\r"
expect "ftp> "
send "put $tarball\r"
expect "ftp> "
send "quit\r"
```

It's as easy to write as it looks. In only a few minutes, you can automate common tasks. To run the script, simply type:

```
$ ./download tarball.tar.gz target
```

where target is the hostname of the target system.

You even can use it to set up configuration files for you (Asterisk has a million of these):

```
spawn telnet $target
send "cd /var/tmp/ipkg/asterisk\r"
expect "> "
send "cp -a etc/* /etc\r"
expect "> "
send "exit\r"
```

This example starts a Telnet session on the target and copies a bunch of files to /etc.

Expect can save you a lot of time in embedded systems work, closing the distance between embedded and regular desktop development.

Plug a regular analog telephone in to one of the FXS ports (the FXS ports will have green LEDs). Pick up the phone, and you will hear a dial tone. Try dialing 2000, and you should hear the "Congratulations, you have..." welcome message. You then can try dialing a few of the demo options and make some calls between extensions.

The above procedure needs to be performed every time you power down or reset the STAMP card. For regular use, the PBX can boot Asterisk from the SD card; however, while experimenting, it is often more convenient to download from the host PC.

The configuration files for Asterisk are the same as for x86 Asterisk. Some useful files are:

- /etc/zaptel.conf: Zaptel driver configuration
- /etc/asterisk/zapata.conf: more Zaptel configuration.
- /etc/asterisk/extensions.conf: sets up dial plan for PBX.
- /etc/asterisk/sip.conf: SIP phone configuration.

These files can be edited with vi on the Blackfin, but remember that any configurations will be lost when the board is powered down or reset. Permanent changes can be stored on the SD card.

Further Work

So, there you have the basic steps for building your own embedded Asterisk PBX. Kits containing the daughterboards and modules are available from the author. We are working on further development—for example, custom Blackfin hardware designed specifically for telephony work and other line interface cards, such as BRI-ISDN. It's an exciting project, with the novel feature of open hardware development.■

David Rowe has 20 years of experience in the development of DSP-based telephony and sat-com hardware/software. David has a wide mix of skills including software, hardware and project management, and a PhD in DSP theory. He has held executive-level positions in the sat-com industry (www.dspace.com.au) and has built and successfully exited a small business (www.voicetronix.com). However, he has decided he is better at debugging machines than people, so he currently chooses to hack telephony hardware and software full time.

Resources

Free Telephony Project:
www.rowetel.com/ucasterisk/index.html

Blackfin uClinux Site: **blackfin.uclinux.org**

Building an Embedded Asterisk PBX, Part 1:
www.rowetel.com/blog/?p=15

Building an Embedded Asterisk PBX, Part 2:
www.rowetel.com/blog/?p=16

Optimizing code for the Blackfin: **www.rowetel.com/blog/?p=5**

gEDA Open EDA Tools: **www.geda.seul.org**

Advertiser Index

VoIP with CommuniGate Pro

How to set up the VoIP features in CommuniGate Pro with a Sipura 3000 and Polycom phone.

DANIEL SADOWSKI AND STEPHEN PRATT

Setting up your CommuniGate Pro (CGP) VoIP phone system is as simple as just installing the program. VoIP functionality is part of the base product, so there is no special configuration or licensing necessary. After you have initially downloaded and installed the platform package of your choice (**www.communigate.com/download**), you simply need to start the CommuniGate Pro server. To do this, at a UNIX prompt on your server, type `/etc/init.d/CommuniGate start`.

Once you have the CommuniGate Pro server up and running, you need to locate the randomly generated postmaster password from the postmaster accounts settings file. For most Linux platforms, the default location is /var/CommuniGate/Accounts/postmaster.macnt/account.settings. `cat` this file to obtain the predefined random postmaster password.

When you have found your postmaster password, open a Web browser of your choice and connect to the CommuniGate Pro Web-Administration Interface at either of these locations (where mail.example.com is the name of your new CGP system in DNS): http://mail.example.com:8010 or https://mail.example.com:9010.

The next step is setting up users on the CGP server. To do this, in the Web-Admin Interface, select the Users tab along the top of the window. CommuniGate Pro will ask you to authenticate—do so as postmaster, with the password you obtained from the postmaster accounts settings file. Next, select the Domains, sub-tab. Select the domain name corresponding to the domain where the new user(s) will be added. Find the button labeled Create Account (Figure 1). Enter the login for a new user in the field to the right of the button. Now, click Create Account.

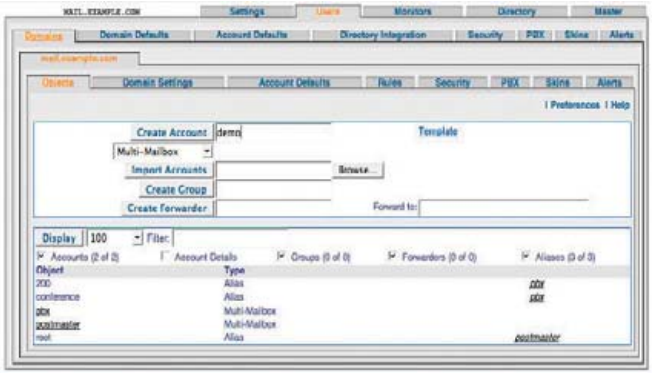


Figure 1. Create the account.

After the account has been created, a Settings page for this new user is displayed. Enter a Real Name for the account, as well as a password. This will be the new user's password for all client applications, such as e-mail, voice and video. Scroll down to the field labeled Aliases. In order to assign an "extension" to the new user, enter the desired extension in the Aliases field. All extensions are really just CGP aliases for an account. Finally, click the Update button located right under the Aliases field. The new user has now been provisioned and already has access to all services, by default, including VoIP.

Repeat the provisioning process above for additional users by clicking the Objects tab located near the top of the tabs. Enter the login for the next new user and so on.

Now that users are provisioned on the CGP server, the next step is to install and configure a device or service known as a VoIP-to-PSTN gateway. The function of such a device or service is to route inbound and outbound telephone calls through the gateway and convert signaling from IP to PSTN and back.

There are numerous PSTN gateway devices supported by CommuniGate Pro, including Cisco Routers with SIP module, AudioCodes Mediant, Mediatrix, VegaStream and Sipura. Also, a number of gateway services are available that CGP supports, such as VoicePulse, Voxeo and Tario.

Suppose, for example, you elect to purchase the Sipura 3000. This device provides decent quality, is well valued and supports three ports:

1. An analog (RJ-11) line from your telephone provider (such as SBC).

2. A LAN Ethernet connection (RJ-45).

3. A second analog (RJ-11) line for connecting a standard telephone.

Note: if you're using a PSTN device such as those listed above, you need to pay for a PSTN line to your location. The smaller devices (such as Sipura and Mediatrix) can use RJ-11 analog lines as input. The larger devices typically require a dedicated T1 PRI to your site.

To configure the Sipura 3000 to work with CGP, follow these steps:

1. Plug in the Sipura—power, analog line (to your wall socket) and Ethernet (to your LAN).

2. By default, the Sipura should get a DHCP address on your network and start its administration interface (accessible via a Web browser).

3. Connect to the administration interface http://IP.address.of.Sipura/admin. The default login is admin, and there is no default password (it should accept a blank password).

The Sipura 3000 uses a tabbed administration interface much like CGP. The Info tab displays the current configuration. The System tab (Figure 2) should be configured for a hostname and any other relevant information to your site (DHCP, Domain, DNS and so forth).

If you are just using the PSTN and Ethernet ports, then the only other configuration changes that need to be set are on the PSTN Line tab. These changes include the following for a very simple setup. Note

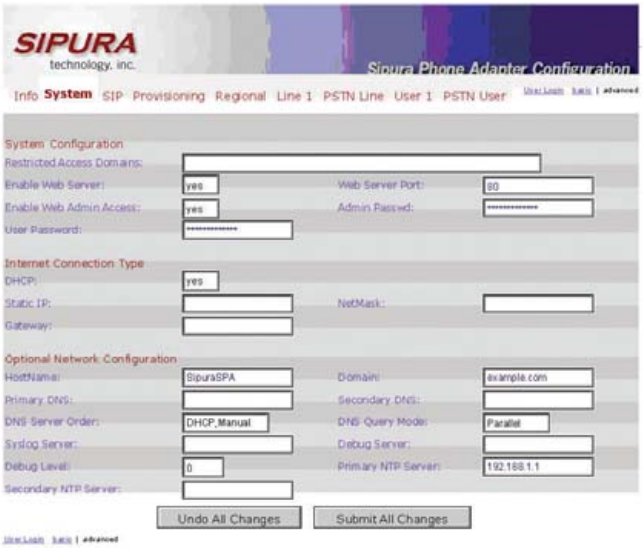


Figure 2. Use the Web interface to set up the Sipura 3000.

that no security restrictions have been added here, so any system that can route SIP traffic to the Sipura device will be able to initiate outbound calls to the PSTN:

```
PSTN Line
Line Enable: yes
NAT Mapping Enable: no
SIP Port: 5060

Proxy and Registration
Proxy: cgpserver.domain.com (fill in correct host)
Use Outbound Proxy: yes
Outbound Proxy: cgpserver.domain.com (or IP address)
Register: no
Display Name: Sipura3000
Use Auth ID: no

Dial Plans
Dial Plan 1: S0<:pbx@cgpserver.domain.com>
Dial Plan 2: (xx.<:@gw0>)


VoIP-to-PSTN Gateway Setup
VoIP-to-PSTN Gateway Enable: yes
VoIP Caller Auth Method: none
One Stage Dialing: yes
VoIP Caller Default DP: 2
```

```
PSTN-To-VoIP Gateway Setup
PSTN-to-VoIP Gateway Enable: yes
PSTN Caller Auth Method: none
PSTN Caller Default DP: 1
```

The dial plans are required with the Sipura device to select a routing destination. Many PSTN gateway devices do not require specific dial plans, as the default VoIP-to-PSTN and PSTN-to-VoIP directions are relatively straightforward. The above dial plans should be entered exactly as listed, except for the hostname of your CommuniGate Pro server. Note too the pbx account name—if you used a different name as the pbx account on your CommuniGate Pro server, it should be used here instead. Now, all incoming calls from the PSTN line will be routed to the LAN network with a destination of this account on the CommuniGate Pro server.

Routing Outbound Calls to the VoIP-to-PSTN Gateway

Routing some or all numeric addresses (calls to the PSTN) from CommuniGate Pro to your FXO Gateway (Foreign eXchange Office—Sipura, in this case) is very easy.

First, log in to the Web-Admin Interface: https://mail.example.com:9010. Then, select the Settings menu option, and then Router (Settings→Router). Next, add the desired routing table entries for the matched numbers you want routed to the FXO gateway. For example:

```
NoRelay:Signal:<1*@example.com> = 1*@192.168.1.136
```

This special syntax simply says, "relay all calls starting with a 1 to the device at 192.168.1.136", which for this example would be a Sipura 3000. Similar routing could be done using a 9 prefix, or 011 for international calls:

```
NoRelay:Signal:<9*@example.com> = *@192.168.1.136
NoRelay:Signal:<011*@example.com> = 011*@192.168.1.136
```

Note: for the 9-prefix example above, the 9 is stripped from the call when it is routed to the FXO device.

Much more complex signal routing and digit-matching plans can be configured—see the on-line CommuniGate Pro guide at **www.communigate.com/CommuniGatePro**.

At this point, you should have your software set up to route calls correctly. Now, you need either a softphone or an IP phone.

CommuniGate Pro supports SIP (Session Initiation Protocol). SIP enables real-time communications, including instant messaging, Voice-over-IP, video conferencing, multimedia, whiteboard and application sharing. Required for implementation is CommuniGate Pro version 5.0 or 5.1 and a SIP-enabled client device.

There are many types of SIP-enabled clients. One type of SIP client is a softphone. A softphone is a voice application run on your desktop computer, laptop computer or mobile computing device that acts just like a normal phone, except that you use a microphone or computer headset to talk.

These clients have all the functionality of their physical counterparts and, in some cases, more. Here are just a few of the softphones available at this time:

- CounterPath's eyebeam and X-Lite products (Windows/Mac).
- Twinkle (Linux).
- SJphone (Windows/Mac/Linux).

Another group of SIP clients, known as IP phones, are physical devices that look just like any other desktop phone. However, these devices are connected to the Ethernet over Internet Protocol instead of the telephone network over twisted pair. These devices are just as easy to set up and configure as their softphone counterparts and typically have an HTML interface for exactly this purpose.

Finally, there are soft-client applications that are capable of providing multiple types of SIP services—including instant messaging, voice, video conferencing and whiteboarding—in one package. Some of these applications include Microsoft Windows Messenger, Linphone and KPhone, with more on the way from both the commercial and open-source sectors.

CommuniGate Systems integrates with all SIP-standards-based IP phones. Phones tested to date include those from Polycom, SNOM, Grandstream, ZyXEL, Hitachi Cable, Cisco and various other lesser-known IP devices.

All SIP-standards phones should work with CommuniGate Pro. However, SIP has evolved and, in the event that a specific device is found not to work, CommuniGate Pro provides a SIP Workaround feature that can be implemented while the system is running to provide specific workarounds to a specific device.

The following describes the setup of a common IP phone, the Polycom 501. Most IP phones use a similar configuration process.

Each Polycom phone contains a built-in Web server for configuration. After putting the phone on your LAN network, it will get a DHCP address, which can be viewed on the phone's LCD screen.

Use your favorite Web browser to connect to the IP address of the phone. The default login for the Polycom phones is:

```
http://IP.address.of.phone
username: Polycom
password: 456
```

After logging in, select the SIP Conf. tab (Figure 3), and enter the hostname or IP address of your CommuniGate Pro system for the Outbound Proxy.

Locate the Registration tab (Figure 4), and enter your Display Name (real name), SIP address (same as your e-mail address) and Auth User ID and Password. The Auth User ID could be your short user name, such as demo, or it also can work as the fully qualified name, such as demo@example.com. Also enter the Address of the server, which in most architectures is the hostname or domain name of your environment, and often it is the same as your Outbound Proxy address above.

That's it. After updating each change, the phone restarts. After your last set of changes, the phone should REGISTER as your account, and your account name will be displayed on the phone's LCD screen. Incoming calls to your account or extension will ring the phone, as well as your other SIP devices.

Figure 3. Set up the Polycom IP phone with the Web interface.

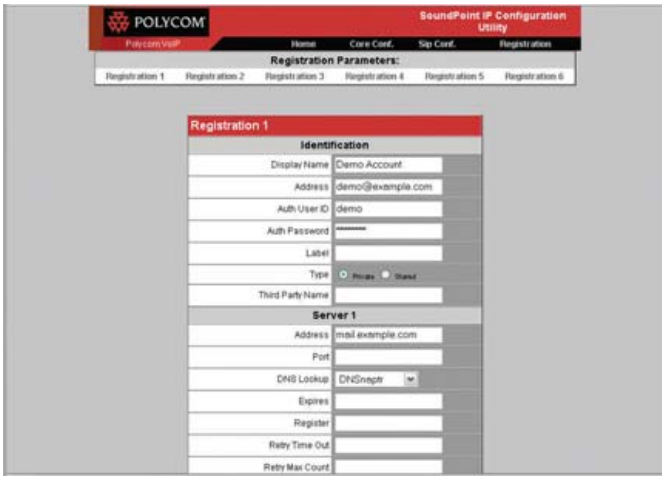


Figure 4. Finish setting up the Polycom phone.

You are now ready to place a call. You can dial from one test user to another you've configured either by entering the login name or, alternatively, the extension assigned as the alias for that user. If you want to call people who also are using VoIP on the Internet, try calling them with their URI. For example, on your eyeBeam softphone, instead of dialing digits, type an address (JohnDoe@another.example.net). Of course, your CGP server must have access to the Internet. Enjoy!■

Daniel Sadowski assists in the marketing department of CommuniGate Systems. He obtained his degree in International Communication Studies from Sonoma State University with study abroad at Uppsala University in Sweden. Sadowski contributes to CGS media relations, in addition to creating and writing of marketing collateral to include case studies, advertising, on-line content and articles.

Stephen Pratt is the Senior Sales and Systems Engineer for CommuniGate Systems. He has extensive experience in messaging servers along with in-depth knowledge of antivirus and antispam and VoIP technologies. His background includes work experience both as employee and installation and services of messaging systems with such companies as SUN Microsystems, Mirapoint, Resumix, Inc., and Computer Curriculum Corporation. Steve also serves on the SPECmail Server subcommittee and helps drive benchmark standards in performance testing.

Building a Multi-Room Digital Music System

Use embedded Linux and open-source software to build a networked audio appliance.

CHAD FILES

Like many media buffs these days, I have a media center computer in my house. It is a small VIA M10000 Mini-ITX system in a Casetronic C158 case running Freevo on top of Gentoo Linux. It sits inside my media cabinet and serves music, video, photos and various other bits of information to my TV. The small media center gets all of its content from a much larger AMD64 Gentoo Linux server that resides in a closet near the back of the house. The two machines talk to each other over a wired network using NFS (Network File System). I will be the first to tell you that it is great to have all of my music, videos and photos in a digital format and easily accessible.

Figure 1. The Media Center with the Nokia 770 as the Controller

In the same spirit of my media center computer, I wanted to have a few small systems that could sit on a shelf or in a drawer that would serve music to different parts of the house. Many products on the market exist that will broadcast audio from a computer and many others will play the resulting stream. The problem with most of these, to me, is that only one stream is available. I wanted the ability to play different music in each room at the same time and control it with one device. A few newcomers to the market do exactly what I wanted—namely the Sonos Digital Music System, which was reviewed in the March 2006 issue of *Linux Journal*. However, I wanted to tackle this one myself.

My idea was to build a fanless Mini-ITX appliance that would grab content from my file server and play it using MPD (Music Player Dæmon). To control the appliance, I use lightTPD (a PHP-enabled Web server) and phpMp (a PHP application that controls MPD), running on the appliance. This gives any computer on the network with a Web browser the power to control the appliance. My controller of choice is a Nokia 770. Using Opera, which comes installed, I can point the browser to each of the appliances and control them over my wireless network.

Hardware for the Appliance

For the appliance, I chose to start with a VIA ML6000EA Mini-ITX motherboard. This board has many features that make it perfect for this type of appliance. The main one is the VIA 600MHz Eden fanless processor. This tiny processor is more than enough to push the applications, and it makes absolutely no noise. Other nice features include six-channel onboard audio, onboard LAN, support for 1GB of memory and a PCI slot.

Because I wanted the appliance to be silent, I decided to forgo a standard hard drive. Instead, I opted for a 256MB Flash drive that plugs directly in to a 40-pin IDE slot. The Flash drive also draws its power from a standard four-pin molex. I also added 256MB of DDR 400 memory to finish out the internals.

Figure 2. The Open Casetronic Case with Motherboard

To give the appliance a sleek look, I went with a Casetronic C158 case. This case has a smooth front, which makes it look more like an appliance. Even though the case has room for a slim ROM drive, I did not put one in, because MPD does not have support for playing audio from a CD. The case also has audio jacks on the front right side that provide easy access for enjoying music with headphones. Another nice feature on this case is the optional CompactFlash card reader that can be mounted from the inside and accessed right below the CD-ROM tray.

Embedded Linux

Because I chose to use a Flash-based hard drive, I needed to make sure that my Linux installation was small, less than 250MB, and did not write to the disk. Flash-based drives and disks have only a certain number of write cycles that can be preformed on them before they give out. My particular drive, made by IEI Global Sourcing, supports more than one-million cycles. Even with support for this many cycles, the drive would stop working in a fairly short amount of time due to disk writes, especially if there were a swap partition on the disk. With all of this factored in, I decided an embedded Linux system was the best fit.

Embedded Linux systems can be created in a variety of ways. I chose to go with embedded Gentoo (see Resources). Because my development machine, a Dell 600m laptop, already runs Gentoo, I didn't need to create a custom build chain. I simply could get a snapshot from the Gentoo Web site and use my laptop as a host to build the embedded system.

Before I attempted to build the system, I took a long look at *Building Embedded Linux Systems* by Karim Yaghmour (see Resources). This book provides a very detailed view of the exact ingredients that make up a functional Linux operating system. And, as the title states, it shows the reader how to build embedded Linux systems. I also followed the instructions written by Heath Holcomb on how to construct an embedded Gentoo system (see Resources). Heath's instructions describe all the needed steps for building an embedded Gentoo system and making it operational. Without these resources, this entire project would not have been possible. I strongly recommend you take a look at them if you are considering doing anything with embedded Linux.

It also should be noted that if you want to build an appliance like this, but would rather use a hard drive and save yourself some time, you can use almost any Linux distribution. Most of them come with all of the software for this project already in their package systems.

Building the embedded Gentoo system was not too difficult. If you have ever installed Gentoo before, making the leap into the embedded realm is not very hard. Several things did make my build interesting though.

I followed Heath's instructions to the letter, and when I was finished, I had a working embedded Gentoo system. After testing it and making sure it all worked, I went on to install the rest of the software that I needed (MPD, lightTPD and phpMp). This proved to be a bit problematic. Portage (Gentoo's package system) wanted to install several software packages that I did not want or need. So, I started over from scratch and carefully installed only the dependencies for the software I needed. Then, I installed my three pieces of software with a nodeps flag to ensure that I did not get the unwanted software.

During the build process, I tested the system several times. I wanted to make sure that with each step I was making progress and not breaking something that had worked previously. To test the system, I used a USB media reader to copy the embedded Gentoo system off my laptop onto a CompactFlash card. I then plugged the CompactFlash card in to a card reader plugged directly in to my appliance computer. This allowed me to test the appliance and emulate how it would operate once everything was working.

Network Setup

As I mentioned earlier, all of my media is on a file server. Having my media stored this way lets me have any number of computers on my network access the files as if they were stored locally. In order to share the files, I set up NFS on the server. Using NFS, I can allow IP addresses, or sets of IP addresses, to connect to specific directories on the server and have specific security rules.

For the appliance computer, I set up two shares. The primary share is where all of the audio content is stored. This share is read-only; I did not want anything mounting this directory to be able to modify it. The second share is a writable share where all of the MPD metadata is stored (more on this later). Both shares are set up with IP restriction only. I decided against using user authentication for reasons of simplicity.

To complete the network setup on the server, I made sure NFS started on boot. Then, I added the mount lines to a startup script on the appliance.

The Music Player Dæmon

At the core of this appliance is MPD. MPD can play almost any audio format: MP3, Ogg, AAC (without DRM), FLAC and so on. For my setup, I enabled Ogg, MP3 and AAC, as all of my music files are in one of these formats. MPD also will play audio streams. This gave my appliance the ability to play all of my favorite Internet radio stations. Configuring MPD is actually fairly easy. Only a handful of settings need to be changed for MPD to function correctly.

The two most obvious settings that need to be changed are music_directory and playlist_directory. I pointed the music directory setting to the first NFS mount (the one with all of the audio files). The playlist directory setting was pointed to a playlist directory on the second NFS mount. By pointing the playlist setting to a common directory, all of the appliances on the network can share playlists.

The next setting I modified was db_file. This setting points to a file where MDP stores all of the metadata about the audio files on the system. I pointed this setting to a file called mpd.db on the writable NFS share. Just like the playlist setting, this allows all the appliances to share a common database of audio information. This also allows any appliance to update the MPD database.

The final setting I modified was state_file. This setting points to a file that holds the information about what state MPD is in when it stops. Again, this points to a file on the writable NFS mount. However, the state file is unique for each appliance. When I turn an appliance back on, I want its state, not the state of another appliance.

Controlling the Appliance

Because the appliance does not have a screen, I needed a way to control the audio playback. Fortunately, the same folks who developed MPD also developed phpMp. phpMp is a small and simple Web application designed to control an instance of MPD using the socket extension in PHP. In order to get phpMp running, I had to install a Web server. Because I was using an embedded platform, I had to be conscious of application size. I also wanted something that was very easy to configure and did not take up too many resources. So, with those considerations in mind, I decided to use lightTPD. lightTPD is a small, fast, full-featured Web server that can run PHP scripts.

Figure 3. Web-Based Control of the Media Center

Out of the box, lightTPD and phpMp needed very little configuration, which is a great feature. Of course, because the appliance is headless, I had to make sure lightTPD started automatically during the boot process.

Problems and Challenges

All things considered, I had very few problems getting everything working. The few issues I did have dealt with the OS being embedded. Fortunately, they were easily remedied.

The first issue was application logging. MPD, lightTPD and the OS itself are configured by default to log everything to files. With my hard drive mounted as read-only, this caused a few issues. To solve the problem, I added a line in the fstab to mount the /tmp directory as a tmpfs. Then, I redirected all logging to the /tmp directory. Doing this allowed all of the applications to log information and still have a read-only filesystem.

The next issue I had dealt with was the ALSA (Advanced Linux Sound Architecture) device files. Most Linux systems nowadays use UDEV to create device files dynamically on boot. Among the device files created by UDEV are the sound card device files that ALSA uses. Because my system did not have UDEV, these files were never created. To solve the issue, I manually created the device files I needed and made them part of my embedded distribution.

The last challenge dealt with turning the appliance off. Because there is no console nor interface to tell the system to shut down, you have to use the power button on the appliance. This led me to building ACPI (Advanced Configuration and Power Interface) support into the embedded system. With ACPI, I configured the system to shut down when the power button is pressed. As the operating system is so small and has so few processes running, shutdown takes only a split second. Eventually, I would like to add support for shutdown to the phpMp interface, but for now, the power button works just fine.

Enhancements and Improvements

Through the course of building this system, I thought of several things I would like to do to enhance the appliance, such as adding an LCD to the front of the case that would show the artist, album and name of the current track. LCDproc has support for MPD, and LCD4linux is adding support, so installing an LCD should be trivial. I also would like to build a small amplifier that could fit into the case so I would not

have to use powered speakers. My ideas were not relegated to the appliance alone; I thought of several enhancements I would like to see to the Nokia 770 and MPD.

The Nokia 770 is the perfect interface for this appliance. As its user base grows, so do the features and the amount of applications it can run. Developers across the world are porting more and more applications to it every day. In the future, I would like to see gmpc (GNOME Music Player Client) ported. gmpc is a GTK+ application that connects to and controls MPD. If gmpc were to be ported to the Nokia 770, lightTPD and phpMp no longer would be needed to control the appliance. The 770 could control it natively.

There also are several enhancements that I would like to see added to MPD to make this appliance even better. The first one is a cover art plugin. Adding support for this has been discussed among the MPD developers, but it has not been added to date. Another nice addition would be a tag editor that allows users to update the meta-information stored in an audio file.

Conclusion

The Mini-ITX line of motherboards, coupled with Linux, provide a solid foundation for building all kinds of appliances like the one described in this article. It took me just short of a month to build and test the appliance. It also was fairly cheap for something so small. All the hardware for the appliance cost about $320 US.

Acknowledgements

I would like to thank my good friend Ryan Corder for helping me along the way. He was instrumental in teaching me the finer aspects of Gentoo and Linux systems as a whole. All of the parts for the appliance were purchased, and shipped very quickly, from Logic Supply (see Resources).■

Chad Files is a software developer who resides in Conway, Arkansas. He is an avid hiker and longtime Linux user. He welcomes your comments at cpfiles@gmail.com.

Resources

Embedded Gentoo: **embedded.gentoo.org**

Building Embedded Linux Systems:
www.oreilly.com/catalog/belinuxsys

Gentoo Embedded x86 Guide:
www.bulah.com/embedded-guide.html

Logic Supply: **www.logicsupply.com**

The D Programming Language

Familiarize yourself with the powerful compiled and managed language D.

AMEER ARMALY

During the past few decades, programming languages have come a long way. In comparison to the dawn of UNIX and C, when compiled languages were just getting their start, today's world is full of languages with all sorts of goals and features. In this article, I discuss one such language, D from Digital Mars. D is a natively compiled, statically typed, multiparadigm C-like language. Its aim is to simplify development of any type of application—from kernels to games—by combining the performance and flexibility of C with the productivity-boosting factors of languages, such as Ruby and Python. It originally was conceived by Walter Bright, the man who wrote the first ANSI C++ compiler for DOS. The reference compiler is freely downloadable for both Windows and Linux, and the front end is licensed under a dual GPL and Artistic license.

GDC is a D compiler, which uses the GCC back end and is distributed under the GPL. D's features include the lack of a preprocessor, a garbage collector, flexible first-class arrays, contracts, inline assembler and more. With all this, it still maintains ABI compatibility with C, allowing you to use all your old C libraries from D easily, with little work. The D standard library includes bindings to all standard C functions as well.

Hello World

In D, the Hello World program goes like this:

```
import std.stdio;  // standard i/o module
int main(char[][] args)
{
        writefln("Hello world!");
        return 0;
}
```

writef is D's typesafe version of printf; writefln adds a newline character at the end. Garbage collector D includes an automatic garbage collector, relieving the programmer of the need to manage memory explicitly. This allows programmers to focus more on the task at hand, as opposed to having to worry about the condition of each memory chunk. Furthermore, it eliminates a whole class of bugs dealing with dangling pointers and invalid memory references. In times when the GC would slow the application down, there is always the option of turning it off altogether or using C's malloc and free for memory management.

Modules

In D, modules are imported using the import statement and have a one-to-one correspondence with source files, with the period as the path separator. Each symbol within a module has two names: the name of the symbol and the name of the symbol prefixed by the module name, which is called the fully qualified name or FQN. For example, writefln can be referred to as writefln or std.stdio.writefln. For cases when the FQN is preferred, the static import statement imports the module's symbols but avoids putting them into the global namespace. For example, both the std.string and std.regexp modules include functions for find, replace and split. Because I'm more likely to use pure string functions than regular expressions, I would statically import std.regexp, so that whenever I wanted to use any of its functions, I would have to be explicit, whereas I simply could call the string functions by their regular names.

Modules can have static constructors and destructors. The static this() function in any module is the static constructor and is invoked before main(); after main has returned the static, ~this() function is invoked. Because modules are imported symbolically, this means there are no header files. Everything is declared once and only once, eliminating the need for declaring functions in advance or declaring classes in two places and trying to keep both declarations consistent.

alias and typedef

In D, there is a distinction made between an alias and a type. A typedef introduces an entirely new type to the type-checking system and to function overloading, which are discussed later. An alias is a simple replacement for a type, or optionally a symbol:

```
alias int size_t;
typedef int myint; //can't implicitly convert to int
alias someReallyLongFunctionName func;
```

Arrays

In D, arrays are first-class types in every way. D contains three types of arrays: static, dynamic and associative arrays. Array declarations read right to left; char[][] is interpreted as an array of arrays of characters:

```
int[] intArray; // dynamic array of ints
int[2][4] matrix; // a 2x4 matrix
```

All arrays have length, sort and reverse properties. Associative arrays are arrays where the index is something other than sequential integers, possibly text strings, structs or arbitrary integers:

```
import std.stdio;
int main(char[][] args)
{
```

```
        int[char[]] petNumber;
        petNumber["Dog"] = 212;
        petNumber["cat"] = 23149;
        int[] sortMe = [2, 9, 341, 23, 74, 112349];
        int[] sorted = sortMe.sort;
        int[] reversed = sorted.reverse;
        return 0;
}
```

Dynamic and static arrays can be sliced with the .. operator. The starting parameter is inclusive, but the ending parameter is not. Therefore, if you slice from zero to the length of an array, you get the whole array:

```
int[] numbers = [1, 2, 3, 4, 5, 6, 7];
numbers = numbers[0..2] // 1-3 now
```

Finally, D uses the ~ operator for concatenation, as addition and concatenation are at their most fundamental two different concepts:

```
char[] string1 = "Hello ";
char[] string2 = "world!";
char[] string = string1 ~ string2; // Hello world!
```

This is a prime example of how D implements a lot of syntactic sugar on top of more low-level routines to make the programmer more focused on the implementation of the task itself. Strings D takes arrays one step further. Because strings are logically arrays of characters, D has no built-in string type; instead we simply declare an array of characters.

Furthermore, D has three types of strings: char, a UTF-8 codepoint; wchar, a UTF-16 codepoint; and dchar, a UTF-32 codepoint. These types, along with standard library routines for manipulating unicode characters, make D a language suited to internationalized programming. In comparison with C, D strings know their length, eliminating even more bugs and security issues dealing with finding the elusive null terminator.

Contracts

D implements techniques that make contract programming easy, which makes for better quality assurance in programs. Making contracts part of the language itself makes it much more likely that they actually will be used, because the programmer doesn't have to implement them or use an outside library for them.

The simplest type of contract is the assert statement. It checks whether the value that is passed to it is true, and if not, it throws an exception. Assert statements can be passed optional message

arguments to be more informative. Functions have two types of contracts, pre and post, signified by the in and out blocks preceding a function. The in contract must be fulfilled before the rest of the function is executed; otherwise, an AssertError is thrown. The post contract is passed the return value of the function and checks to make sure the function did what it was supposed to do before the value is passed to the application. When a program is compiled with the release option turned on, all asserts and contracts are removed for speed purposes:

```
int sqrt(int i)
in {
        assert(i > 0);
   }
out(result) { // the return value is
              // always assigned to result
assert((result * result) ==i);
}
body
{...}
```

Another type of contract is the unit test. Its purpose is to ensure that a particular function or set of functions is working according to specification with various possible arguments. Suppose we have the following rather useless addition function:

```
int add(int x, int y) { return x + y; }
```

The unit test would be placed in the same module, and if the unittest option is enabled, it would be run as soon as the module is imported and any function from it is executed. In this case, it probably would look something like this:

```
unittest {
        assert(add(1, 2) == 3);
        assert(add( -1, -2) == -3);
}
```

Conditional Compilation

Because D has no preprocessor, conditional compilation statements are part of the language itself. This removes the numerous headaches caused by the preprocessor along with the infinite ways in which it can be used, and it makes for a faster compile. The version statement is a lot like #ifdef in C. If a version identifier is defined, the code under it gets compiled in; otherwise, it doesn't:

```
version(Linux)
import std.c.linux.linux;
else version(Win32)
import std.windows.windows;
```

The debug statement is a lot like the version statement, but it doesn't necessarily need an identifier. Debugging code can be placed in the global debug condition or in a specific identifier:

```
debug writefln("Debug: something is happening.");
debug (socket) writefln("Debug: something is
happening concerning sockets.");
```

The static if statement allows for the compile-time checking of constants:

```
const int CONFIGSOMETHING = 1;

void doSomething()
{
        static if(CONFIGSOMETHING == 1)
        { ... }
}
```

Scope Statement

The scope statement is designed to make for a more natural organization of code by allowing a scope's cleanup, success and failure code to be logically grouped:

```
void doSomething()
{
    scope(exit) writefln("We exited.");
    scope(success) writefln("We exited normally.");
    scope(failure) writefln("We exited due to an exception.");
        ..
}
```

Scope statements are executed in reverse order. Script syntax DMD, the reference D compiler, supports the -run option, which runs the program taken from standard input. This allows you to have self-compiling D scripts, as long as the appropriate line is at the top, like so:

```
#!/usr/bin/dmd -run
```

Type Inference

D allows the automatic inferring of the optimal type of a variable with the auto-declaration:

```
auto i = 1; // int
auto s = "hi"; // char[4]
```

This allows the compiler to choose the optimal type when that functionality is needed.

foreach

Some of you might be familiar with the foreach construct; it essentially says, "do this to every element of this array" as opposed to "do this a set number of times, which happens to be the length of the array". foreach loops simplify iteration through arrays immensely, because the programmer no longer even has to care about the counter variable. The compiler handles that along with making each element of the array available:

```
char[] str = "abcdefghijklmnop";
foreach(char c; str)
writefln(c);
```

You also can obtain the index of the element by declaring it in the loop:

```
int [] y = [5, 4, 3, 2, 1];
foreach(int i, int x; y)
writefln("number %d is %d", i, x);
```

Finally, you can avoid worrying about the types of the variables, and instead use type inference:

```
foreach(i, c; str)
```

This opens up the field for numerous compiler optimizations that could be performed—all because the compiler is taking care of as much as possible while still providing the programmer with the flexibility to accomplish any given task.

Exceptions

As a rule, D uses exceptions for error handling as opposed to error codes. D uses the try-catch-finally model for exceptions, which allows cleanup code to be inserted conveniently in the finally block. For those cases when the finally block is insufficient, scope statements come in quite handy.

Classes

Like any object-oriented language, D has the ability to create object classes. One major difference is the lack of a virtual keyword, unlike with C++. This is handled automatically by the compiler. D uses a single-inheritence paradigm, relying on interfaces and mixins, which are discussed later to fill in the gaps. Classes are passed by reference rather than by value, so the programmer doesn't have to worry about treating it like a pointer. Furthermore, there is no -> or :: operator; the . is used in all situations to access members of structs and classes. All classes derive from Object, the root of the inheritance hierarchy:

```
class MyClass {
        int i;
        char[] str;
        void doSomething() { ... };
}
```

Classes can have defined properties by having multiple functions with the same name:

```
class Person {
        private char[] PName;
        char[] name() {return PName;}
        void name(char[] str)
        {
        // do whatever's necessary to update any
        // other places where the name is stored
        PName = name;
        }
}
```

Classes can have constructors and destructors, namely this and ~this:

```
class MyClass {
        this() { writefln("Constructor called");}
        this(int i) {
          writefln("Constructor called with %d", i);
        }
        ~this() { writefln("Goodbye");}
```

Classes have access to the constructors of their base class:

```
this(int i) {
        super(1, 32, i); // super is the name of the
                         // base class constructor
}
```

Classes can be declared inside other classes or functions, and they have access to the variables in that scope. They also can overload operators, such as comparison, to make working with them more obvious, as in C++.

Classes can have invariants, which are contracts that are checked at the end of constructors, before destructors and before public members, but removed when compiling for release:

```
class Date
{
int day;
int hour;
invariant
    {
        assert(1 <= day && day <= 31);
        assert(0 <= hour && hour < 24);
    }
}
```

To check whether two class references are pointing to the same class, use the is operator:

```
MyClass c1 = new MyClass;
MyClass c2;
if(c1 is c2)
        writefln("These point to the same thing.");
```

Interfaces

An interface is a set of functions that any class deriving from it must implement:

```
interface Animal {
        void eat(Food what);
        void walk(int direction);
        void makeSound();
}
```

Functions

In D, there is no inline keyword—the compiler decides which functions to inline, so the programmer doesn't even have to worry about it. Functions can be overloaded—that is to say, two functions with the same name can take different parameters, but the compiler is smart enough to know which one you're talking about:

```
void func(int i) // can implicitly take
                 // longs and shorts too
{...}

void func(char[] str)
{...}

void main()
{
        func(23);
        func("hello");
}
```

Function parameters can be either in, out, inout or lazy, with in being the default behavior. Out parameters are simple outputs:

```
void func(out int i)
{
        I += 4;
}

void main()
{
        int n = 5;
        writefln(n);
        func(n);
        writefln(n);
}
```

inout parameters are read/write, but no new copy is created:

```
void func(inout int i)
{
        if(i >= 0)
                ..
        else
                ..
}
```

Lazy parameters are computed only when they are needed. For example, let's say you called a function like this:

```
log("Log: error at "~toString(i)~" file not found.");
```

Notice that every time you call it, the strings are concatenated and passed to the function. The lazy storage class means that the strings are put together only if they are called upon, increasing performance and efficiency. Nested functions in D allow the nesting of functions within other functions:

```
void main()
{
        void func()
        {
                ..
        }
}
```

Nested functions have read/write access to the variables of the enclosing function:

```
void main()
{
int i;
        void func()
        {
                writefln(i + 1);
        }
}
```

Templates

D has a totally redesigned and highly flexible template system. For starters, the ! operator is used for template instantiation. This eliminates the numerous ambiguities caused by <> instantiation and is more readily recognizable. Here is a simple copier template:

```
template TCopy(t) {
        void copy(T from, out T to)
```

```
        {
                to = from;
        }
}

void main()
{
        int from = 7;
        int to;
        TCopy!(int).copy(from, to);
}
```

Template declarations can be aliased:

```
alias TFoo!(int) temp;
```

Templates can be specialized for different types, and the compiler deduces which type you are referring to:

```
template TFoo(T)         { ... }  // #1
template TFoo(T : T[])   { ... }  // #2
template TFoo(T : char)  { ... }  // #3
template TFoo(T,U,V)     { ... }  // #4
alias TFoo!(int) foo1;            // instantiates #1
alias TFoo!(double[]) foo2;       // instantiates #2
                                  // with T being double
alias TFoo!(char) foo3;           // instantiates #3
alias TFoo!(char, int) fooe;      // error, number of
                                  // arguments mismatch
alias TFoo!(char, int, int) foo4; // instantiates #4
```

Function Templates

If a template has one member function and nothing else, it can be declared like this:

```
void TFunk(T) (T i)
{
..
}
```

Implicit Function Template Instantiation

Function templates can be instantiated implicitly, and the types of the arguments deduced:

```
TFoo!(int, char[]) (2,"foo");
TFoo(2, "foo");
```

Class Templates

In those cases when you need to declare a template and its only member is a class, use the following simplified syntax:

```
class MyTemplateClass (T)
{
..
}
```

Mixins

A mixin is just like cutting and pasting a template's code into a class; it doesn't create its own scope:

```
template TFoo(t)
{
        t i;
}

class test
{
        mixin TFoo!(int);
        this()
        {
                i = 5;
        }
}

void main()
{
        Test t = new Test;
        writefln(t.i);
}
```

Conclusion

D is a promising language that is able to supply many different needs in the programming community. Features such as arrays, SH syntax and type inference make D comparable to languages, such as Ruby and Python, in those regards, while still leaving the door open for low-level system programmers with the inline assembler and other features. D can be applied to many programming paradigms—be they object-oriented with classes and interfaces, generic programming with templates and mixins, procedural programming with functions and arrays, or any other. The garbage collector relieves the programmer of the need to manage all memory chunks manually, while still making it possible for those situations in which manual memory management is preferred. It brings in features of imperative languages, such as Lisp, with the lazy storage class, which drastically speeds up efficiency. The language is relatively stable, with the occasional new features or changes added in.

In short, D is a language ready for real-world deployment.■

Ameer Armaly is a blind 18-year-old high-school senior. Among his interests are playing the guitar, programming and science fiction.

Resources

D Specification and Reference Compiler: **www.digitalmars.com/d**

GDC: **dgcc.sf.net**

Numerous Open-Source Projects and Tutorials: **www.dsource.org**

Dealing with the Devil

Who sings the praises of those who got rich taking bribes from Al Capone?

Nick Petreley, Editor in Chief

Those who fail to learn from history are doomed to repeat it. Novell has failed to learn from history, and is doomed to repeat it after having made a deal with Microsoft.

Before I continue, consider that I am one of the most pragmatic and therefore least religious among Linux advocates. I see nothing wrong with good proprietary commercial software from reliable sources. I think the DCMA, not TiVo, is the problem when it comes to TiVoization. Let me add that when IBM behaved as a controlling monopolist that did not work in the best interests of its customers, I repeatedly blew the whistle on IBM's monopolistic business practices. (Obviously, I've been writing for a long time.) Now I am a fan of IBM. As with IBM, my criticism of Microsoft is not personal. If Microsoft ever changes its ways of doing business, as IBM did, I will gladly support the company. But Microsoft has shown no signs that it has abandoned its Machiavellian tactics.

My point is that the following is not the rantings of a free software or Linux zealot, nor is the following the rantings of a mindless Microsoft basher. It is simply the truth, as best as I can see it.

Microsoft is an enemy of Linux, if not *the* enemy. Anyone who thinks otherwise needs to read Steve Ballmer's statement that everyone who runs Linux owes money to Microsoft. But it's worse than that. Microsoft is a company run by megalomaniacs bent on control and domination of every market the company enters. Therefore, every time Microsoft makes a deal, it makes it with the ultimate goal of marginalizing the competition—not destroying all competition, lest it risk another bout with DOJ, but marginalizing it enough to prevent it from posing a threat to its dominance and market leverage. For every deal Microsoft makes that seems beneficial to anyone but Microsoft, you will find an ulterior motive in line with the aforementioned goal. For example, when you peel back the layers of Microsoft's philanthropic deals for Windows, you'll find the internal memo that instructs sales to offer such gestures only where Linux is a threat.

History is littered with the battered bodies left from such deals. I find it somewhat poetic that I first heard the term "coopetition" at a Novell press event, long ago. Novell announced that it made a deal with Microsoft, and then subjected editors to such an overdose of the words "coopetition" and "partnering", it is still painful to write these words today. I don't recall the date of the event, but it was roughly 1–3 years before Windows NT essentially buried NetWare. Granted, Microsoft probably offered as an alternative the refusal to make IPX work properly with Windows 95. But Novell was gutless, all the same, and paid for its mistake.

There isn't enough room to document all the deals Microsoft made that marginalized its competition, so I'll leave it to students of history to recount them. Put simply, nobody has ever walked away from a deal with Microsoft as a winner, at least not in the sense of a true winner or long-term winner.

I've drawn a comparison between Microsoft and Al Capone before. The last time I did so, I quoted from the movie *The Untouchables* as to how to deal with Microsoft. Malone (Sean Connery) says to Elliot Ness (Kevin Costner), "Here's how you get Capone: he pulls a knife, you pull a gun. He sends one of yours to hospital, you send one of his to the morgue! That's the Chicago way! And that's how you get Capone. Now, do you want to do that? Are you ready to do that?"

The last time I quoted that passage in a column, Marc Andreesen wrote to tell me he taped that column to his bathroom mirror to remind him daily how to "negotiate" with Microsoft. I don't know if it did any good, but I hope it helped give Netscape the courage to go to the DOJ.

Some people are content with the Microsoft/Novell deal because Microsoft is going to promote SUSE, and Novell stands to make money from it, at least in the short term. But when it comes to the story of Al Capone, whose praise do we sing? Do we honor as heroes the men who got rich taking bribes from Capone? Or do we honor as heroes the Untouchables, those who were not willing to profit from a deal with the devil?

To those who think the Microsoft/Novell deal is good for Novell and for Linux, I say let's revisit this issue in five years when the current deal expires. I suspect the ulterior motives will surface before then, but if not, I predict those who praise Novell and/or Microsoft will be in for a rude awakening five years from now. I hope I'm wrong, but if I'm right, I pray someone at Novell with some guts will post this column on his or her bathroom mirror before it is too late.■

Nicholas Petreley is Editor in Chief of *Linux Journal* and a former programmer, teacher, analyst and consultant who has been working with and writing about Linux for more than ten years.

Rackspace – Managed Hosting Backed by **FANATICAL SUPPORT™**

Fast servers, secure data centers and maximum bandwidth are all well and good. In fact, we invest a lot of money in them every year. But we believe hosting enterprise class web sites and web applications takes more than technology. It takes Fanatical Support.

Fanatical Support isn't a clever slogan, but the day to day reality our customers experience working with us. It's how we have reimagined customer service to bring unprecedented responsiveness and value to everything we do for our customers. It starts the first time you talk with us. And it never ends.

Contact us to see how Fanatical Support works for you.

1.888.571.8976 or visit **www.rackspace.com**

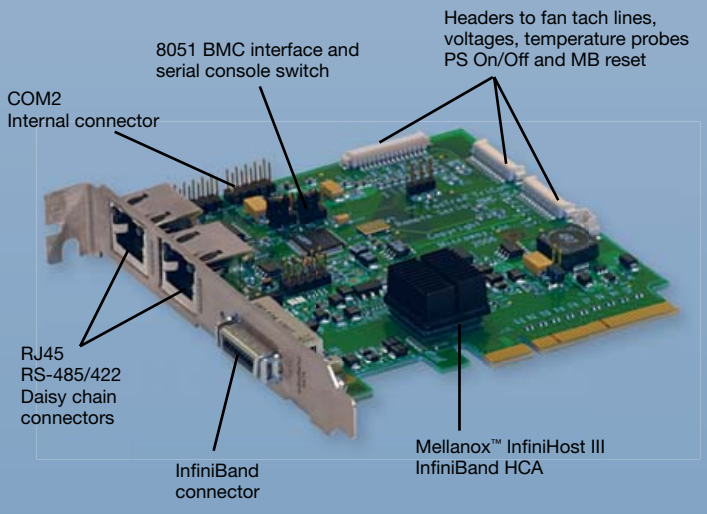
rackspace®
MANAGED HOSTING

Affordable InfiniBand Solutions

4 Great Reasons to Call Microway NOW!

1 TriCom™

- DDR/SDR InfiniBand HCA
- "Switchless" serial console
- NodeWatch web enabled remote monitor and control



2 FasTree™

- DDR InfiniBand switches
- Low latency, modular design
- 24, 36 and 48 port building blocks

3 ServaStor™

- Extensible IB based storage building blocks
- Redundant and scalable
- Parallel file systems
- Open source software
- On-line capacity expansion
- RAID 0,1,1E, 3, 5, 6, 10, 50

4 InfiniScope™

- Monitors ports on HCA's and switches
- Provides real time BW diagnostics
- Finds switch and cable faults
- Lane 15 interface
- Logs all IB errors

Upgrade your current cluster, or let us design your next one using Microway InfiniBand Solutions.

To speak to an HPC expert call ***508 746-7341*** *and ask for technical sales or email*
sales@microway.com
www.microway.com

Microway®
Technology you can count on℠